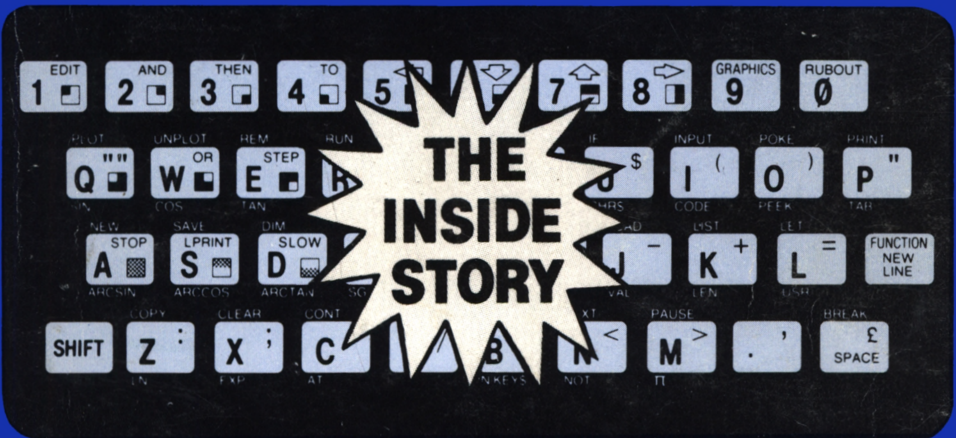
THE ZX81 COMPANION



Bob Maunder

LINSAC

The ZX81 Companion

by

Bob Maunder

ISBN 0 907211 01 1.

LINSAC, 68 Barker Road, Middlesbrough TS5 5ES

© LINSAC 1981

FIRST EDITION
First Printing . . . . July 1981
Second Printing . . . . . . November 1981
Third Printing . . . . March 1982

All Rights Reserved. No part of this publication may be reproduced or transmitted by any means without the prior permission of LINSAC.

Cover picture reproduced from the Sinclair ZX81 Manual.

Printed by Prontaprint, Middlesbrough

PREFACE

The Sinclair ZX81 microcomputer has been widely acclaimed as a tremendous breakthrough in personal computing, even surpassing its predecessor the ZX80. Certainly no other computer has been bought in such quantities by such a wide range of people in such a short space of time since its launch in February 1981. The ZX81 advertising campaign has sought to attract the general public to the concept of using a computer in the home.

"The ZX81 Companion" has been written to assist ZX81 owners in using their computer in the specific areas of information retrieval, education, and games. The Sinclair ZX81 Manual, while being an excellent introduction to ZX81 BASIC, does not discuss any real uses for the machine. However in the **Companion**, readers will find documented programs that can be used immediately to utilise the ZX81 to its full potential, as well as detailed guidelines on the design and development of their own programs. The book is therefore aimed at those familiar with the concepts of ZX81 BASIC but keen to get the ZX81 moving onto higher things. The fourth chapter is aimed at more advanced users who are interested in the workings of the ZX81 Monitor and methods of displaying and using Monitor routines.

It is the opinion of the author that for any serious applications the ZX81 definitely requires the addition of a 16K RAM pack. However many programs in the book can be run on 1K machines, the main exception being Chapter Two which develops a sophisticated information retrieval package for which 16K is naturally vital.

The author has been involved in the ZX series of microcomputers since he acquired the first ZX80 kit in March 1980, and he is co-author of Linsac's 'The ZX80 Companion'. He holds an MSc in Computer Science from Birmingham University and is Head of Computing at Hartlepool College, where he pioneered the use of the ZX80 in education.

Thanks are due to Sinclair Research for permission to reprint the ZX81 keyboard layout (but not the Monitor listing!), to Joe Foster for contributing the Appendix on program development and to Ian Logan for the section on Monitor routines and entry points.

CONTENTS

INTRODUCTION AND NOTATION

Readers who own 16K ZX81's will be able to get the most out of this book, but those with 1K ZX81's or updated ZX80's will also benefit. Memory requirements for programs are clearly marked, and in many of the routines in Chapter One in particular, 1K and 16K alternatives are given. It is the author's opinion that owners of 8K ROM ZX80's are certainly at a disadvantage with regard to the main benefit of the ZX81 – animated displays. Such users will be well advised to consider the purchase of a conversion kit, currently available in the UK from Compshop Ltd., to provide the SLOW compute and display facility. However ZX80 owners without this conversion will still be able to use most of the programs herein, in some cases with the addition of suitable PAUSE statements to simulate SLOW mode.

Material in the four chapters is developed from a simple starting point, and in the first three chapters exercises are used to give the reader practice in the techniques discussed. Solutions are found at the end of the book.

A technique known as logical assignment is used in many of the programs to save on program space: a necessity for 1K machines. This technique combines several conditions and values in a single LET statement, and may not be familiar to some readers: study of the Sinclair Manual is recommended to clarify the use of the technique.

The notation used in the program listings is designed to be as unambiguous as possible. Since spaces in printed text can be important in some circumstances, many of the listings specify a space by the letter b (for blank). Confusion between the letter I and the number 1, or the letter O and the number zero can occur so the following conventions are used:

I	=	letter
1	=	number one
O	=	letter
Ø	=	number zero

Graphics and inverse characters can also be difficult to represent. If text is to be represented in inverse form then this is indicated by the word "inverse" in brackets at the end of the PRINT statement. Graphics characters are generally drawn in and sometimes also identified by their key, e.g. 5ØØ PRINT "■" (inverse space)

CHAPTER ONE
GRAPHICS AND REALTIME TECHNIQUES

1.1 INTRODUCTION

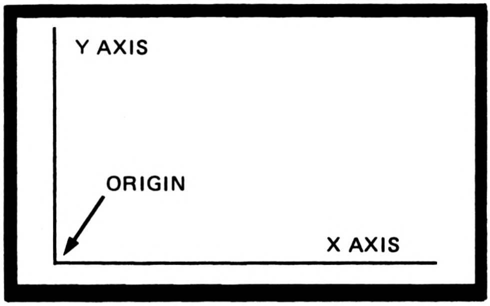
We consider in this chapter the use of ZX81 statements to produce diagrams, pictures and moving displays. **Graphics** is the art of drawing items on the ZX81 screen by means of addressing different parts of the display as you might fill in squares on a piece of graph paper. **Realtime** methods involve getting the ZX81 to respond to you *immediately* : although all ZX81 programs work in a conversational mode with the user entering information (in response to INPUT statements) and the computer replying with a display, programs can be written which will react immediately the user presses a key, whether or not the computer was doing something else at the time.

These two techniques can be immensely useful. On the serious side, information can often be more clearly presented and understood if it is in the form of diagrams, such as graphs or histograms; simple maps or room layouts can also be shown. On the lighter side, games have much more realism and challenge if they involve pictures, and if the pictures move and the player has to respond quickly to this movement, so much the better.

It will be helpful if the reader has looked over Chapters 17, 18 and 19 of the'ZX81 BASIC Programming' manual first. The statements covered in the theory and practical exercises below are PLOT, UNPLOT and PRINT AT (graphics) and INKEY$ and PAUSE (realtime). Do not be deterred by the initial emphasis on theory: in order to produce good graphics you need to have a good grasp of what is often titled 'coordinate geometry'. At the end of this chapter you will be programming your own arcade-type games so stick with it!

1.2 AXES AND COORDINATES

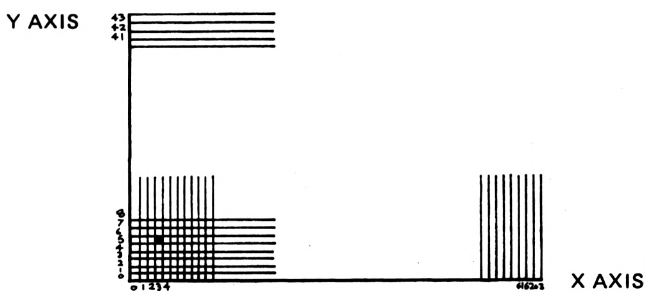
In using the graphics features of the ZX81 we think of the TV screen as a piece of graph paper split into squares. We can black-in a square using PLOT and rub out a blacked-in square using UNPLOT. However to pick out a blacked-in square we must have some way of identifying squares to the ZX81, and this is done by considering the screen as having two lines of reference or axes, at right angles to each other at the left and bottom of the screen.



The vertical axis at the left of the screen is known as the **y axis**, and the horizontal axis at the bottom is called the **x axis**. The point at which they intersect is called the **origin**.

Coordinates

The number of 'squares' on the ZX81 screen is fixed at 64 x 44, i.e. there are 64 divisions along the x axis and 44 divisions along the y axis. To complicate the issue the divisions are numbered from 0 to 63 and from 0 to 43, as shown below.



To identify a particular square on the graph we specify how far along the x axis it is, and then how far along the y axis. For example the blacked-in square in the diagram above is at position 3 on the x axis and position 5 on the y axis and we say its position on the graph is therefore (3,5). This pair of numbers in brackets is known as the coordinates of the square. Note that the Sinclair Manual calls these squares "pixels". The PLOT statement uses the coordinates to identify a square's position and black it in (however brackets are omitted). Try this:

```
PLOT 3, 5
```

A black square appears towards the bottom left hand corner of the screen, or at position (3,5).

Any square in the 64 x 44 graph can be identified using coordinate pairs from the origin at (Ø,Ø) to the top right at (63,43). RUN the following program to get the four corners of the screen display

```
1Ø PLOT  Ø, Ø
2Ø PLOT  Ø, 43
3Ø PLOT  63, Ø
4Ø PLOT  63, 43
```

The next section shows how squares may be drawn in groups to form lines.

1.3 STRAIGHT LINES

Equations of X and Y Axes

A straight line may be drawn on the screen by drawing in several squares together. The squares which form a line all have something in common and we can form an equation for a line using this fact. As an example, consider squares along the x axis:

(Ø, Ø) , (1,Ø) , (2, Ø) and so on up to (63, Ø)

All of these squares have something in common – they have their y position equalling zero. Therefore we say that the x axis has the equatio

y = Ø

Similarly all squares along the y axis have their x coordinate equalling zero so the equation of the y axis is

x = Ø

Therefore in order to draw in the y axis on the screen, all we need to do is PLOT every square where x = Ø. Thus

```
10 FOR Y = Ø TO 43
20 PLOT Ø,Y
30 NEXT Y
```

Add the following lines and we produce a set of x and y axes on the screen.

```
40 FOR X = Ø TO 63
50 PLOT X,Ø
60 NEXT X
```

In fact any vertical line will have an equation

x = a number

while any horizontal line will have an equation.

y = a number

Drawing a Rectangle

You can get some interesting visual effects using just these simple concepts. The following program draws the edges of the screen 'graph':

```
10 FOR X = Ø TO 63
20 PLOT X,Ø
30 PLOT X,43
40 NEXT X
50 FOR Y = Ø TO 43
60 PLOT Ø, Y
70 PLOT 63,Y
80 NEXT Y
```

Notice how the vertical lines and horizontal lines are plotted in pairs through use of a pair of PLOT statements in each of the two loops. For 1K ZX81's, substitute 37 for 43 in lines 3Ø and 5Ø for a complete rectangle.

Another example shows how the entire screen may be blacked in from the left

```
10 FOR X = 0 TO 63 . . . . . (Use 61 for 1K ZX81's)
20 FOR Y = 0 TO 43
30 PLOT X,Y
40 NEXT Y
50 NEXT X
```

Try reversing the order of the loops and see the effect.

Exercise 1 (a): Produce an entirely black screen display by drawing vertical lines from right to left, going down the screen.

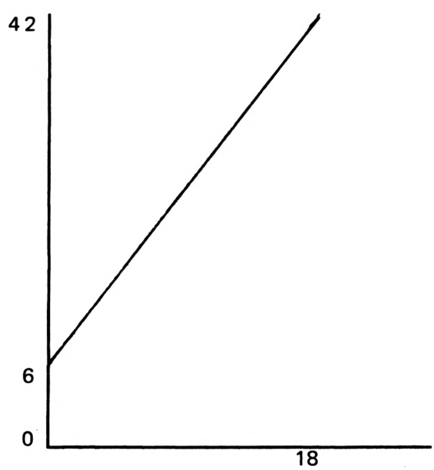
1 (b): Draw a black square with its bottom left corner at position (10,5), sized 20 x 20 squares.

(Solutions on page 115)

Equations of General Lines

Most lines that we will need to draw on the ZX81 will not be vertical or horizontal, but diagonal. We now discuss how we can work out the common features or **equations** of such lines, and thus how they can be plotted on the screen.

The diagram below shows a line drawn between points (0,6) and (18,42).



If this line were drawn on graph paper we would see that it also passes through a sequence of positions starting

(1,8) (2,10) (3,12) (4,14) (5,16) ...

The common factor about all the positions through which the line passes is that the y coordinate is twice the x coordinate plus six. We can therefore say that the line has the following equation

$$y = 2x + 6$$

and we can therefore draw it on the ZX81 screen thus

```
10 FOR X = 0 TO 63
20 PLOT X, 2*X + 6
30 NEXT X
```

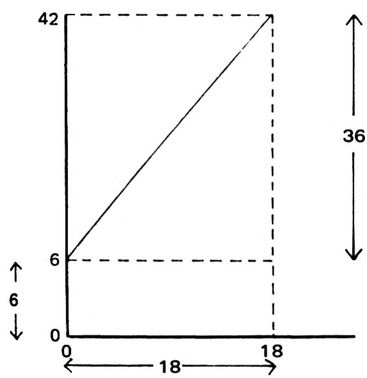
However this teminates with error code B after getting as far as x = 18, since the y value calculated when x = 19 is 44, which is off the screen.

Any diagonal line that we care to choose can be reduced to a simple equation of the form

$$y = mx + c$$

where m and c represent numbers.

The values of m and c can be seen more clearly from the following graph showing y = 2x + 6 again.



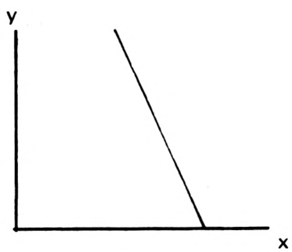
e gradient or steepness of the line is measured by height divided by ıgth. As shown above the line goes *up* 36 squares as it goes *along* 18 ıares, so the gradient is 36 ÷ 18 or 2. This represents m in the general ıation of a straight line, y = mx + c. Similarly c is given by where the e cuts the y axis. To understand this, remember that the y axis is where = Ø. Therefore when the line y = mx + c and the line x = Ø intersect en

$$y = m.\emptyset + c$$

$$= y = c$$

is value is often called the y intercept.

nsider a different line. This one slopes downwards and cuts the x-axis.



ain this line fits the general equation y = mx + c, but this time the dient m will be negative. The x intercept is easily found by remem-ing that the x-axis is where y = Ø.

so y = mx + c
becomes Ø = mx + c
therefore $x = \frac{-c}{m}$

e following program can be used to demonstrate the effects of different ues for m and c.

```
 5 REM ENTER VALUES AND PRINT EQUATION
1Ø CLS
2Ø PRINT "M=";
3Ø INPUT M
4Ø PRINT M;"bC=";        (b = space)
```

```
 50 INPUT C
 60 CLS
 70 PRINT AT 0,12;"Y=";M;"X+";C
 75 REM DRAW AXES
 80 FOR X=0 TO 63
 90 PLOT X,0
100 NEXT X
110 FOR Y=0 TO 43
120 PLOT 0, Y
130 NEXT Y
135 REM DRAW LINE
140 FOR X=0 TO 63
150 PLOT X,M*X+C
160 NEXT X
```

(For 1K ZX81's omit the REM lines)

RUN the program with varying positive and negative values for m and c and finally c = 0 or m = 0. If the y value becomes negative some peculiar effects occur because the PLOT statement always takes the positive values of coordinates. To overcome this add the line

```
145 IF M*X+C<0 THEN STOP
```

If you find it difficult to understand why the ZX81 does not do **continuous** diagonal lines as it would if they were horizontal or vertical, remember that it is only blacking-in squares on a grid. You may have come across computers which appear to draw continuous lines on an output screen, but this is only because the number of squares or **resolution** of the display is higher.

1.4 MOVING OBJECTS

Moving Spots

To relieve what for some might be a tedious excursion into school maths, let us look at how we can get things to move on the ZX81 screen.

The way to produce animation by computer is the same as in cartoons: display a picture then display it in a slightly different position, and so on. When we drew lines on the screen we saw them being extended, and all we need to do to get moving spots is to rub out the trail. Modifying one of the previous routines gives us

```
10 FOR X=0 TO 63
20 PLOT X,0
30 UNPLOT X,0
40 NEXT X
```

and we see a spot moving quickly along the bottom of the screen. To slow it down and make it a bit clearer we could add a PAUSE

```
25 PAUSE 10                  (Vital for updated ZX80's)
26 POKE 16437,255
```

Two points here – remember to include the POKE after every PAUSE if you use FAST mode, and also make sure you PAUSE while the spot is on the screen, not after you have just rubbed it out.

Exercise 1(c): Write a program to get a spot to move round the edges of the screen anti-clockwise starting at the origin (the program above starts you off).

Moving Objects

For greater realism a complete object can be built up and moved across the screen. As we will see later this is much better using the PRINT AT instruction but it can be achieved by PLOT, as below

```
 10 FOR X=0 TO 61
 20 PLOT X,0
 30 PLOT X+1,0
 40 PLOT X+2,0
 50 PLOT X+1,1
 60 PAUSE 10         )   or try  60 FOR A=1 TO 20
 70 POKE 16437,255   )           70 NEXT A
 80 UNPLOT X,0
 90 UNPLOT X+1,1
100 NEXT X
```

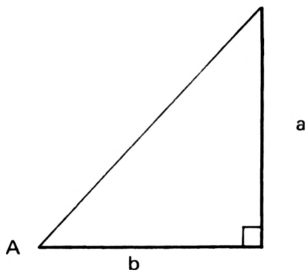
We see that not all of the object need be rubbed out each time, since the remaining part forms part of the next drawing of the object. The annoying blinking is much less accentuated using PRINT AT as we shall see, or even using the dummy loop.

1.5 TRIGONOMETRY

Tangents

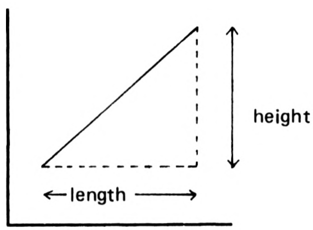
If you have started to read this section in spite of seeing the title then you are doing well. It is true that sines, cosines and particularly tangents can be useful in our theory of graphics. We will consider just tangents, but you can read up any secondary school maths text book to swot sines and cosines if you find it interesting.

A tangent is the ratio of two sides of a right-angled triangle:



The tangent of the angle at A is $\frac{a}{b}$

But think of this on a graph



and we see that the tangent is the same as the gradient of a straight line.

Therefore we can start talking about lines being drawn at certain angles on the screen. For example the following program invites you to enter an angle (0-90°) and it then draws a line from the origin at this angle to the x axis.

```
10 PRINT AT 0,0;"ANGLE=";
20 INPUT A
30 PRINT A
40 IF A<0 OR A>90 THEN GO TO 20
```

```
 5Ø LET M=TAN(A*2*PI/36Ø)
 6Ø FOR X=ØTO 63
 7Ø IF M*X>43 THEN GO TO 1ØØ
 8Ø PLOT X,INT(M*X)
 9Ø NEXT X
1ØØ PAUSE 1ØØ
11Ø POKE 16437,255
12Ø PRINT AT Ø,6;"bbbb"
13Ø GO TO 1Ø
```

(b = one space)

ıes 10–50 invite the user to enter an angle and then the value of m in ə general equation for straight lines through the origin (y = mx) is lculated. As an added complication, the ZX81 will only handle ıgents of angles expressed in **radians** which is a unit of circular measure t since one degree is $\frac{2\pi}{360}$ radians (or $\frac{\pi}{180}$)

: can do an easy conversion.

ıes 60 – 90 draw the line, making sure to stop drawing when the top of the screen is hit

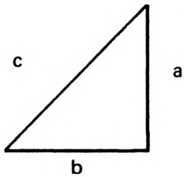
ıes 100 – 130 cause the program to repeat so that several lines can be drawn on the same graph.

Exercise 1 (d): Write a program to draw a "spider's web" of lines, similar to the ones above using the angles 0° to 90° at 5° intervals.

ıe program above and the exercise will crash when the angle is equal to ıety degrees because the tangent of 90° is infinitely large – draw the angle if you cannot see why! A good way of stopping it anyway!

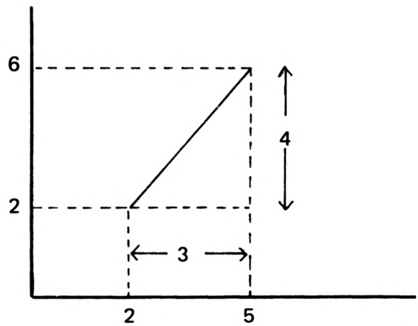
Pythagoras

ıe above-named gentleman may again not be too popular amongst ne readers but his theorem can help us draw some nice pictures if thing else. Basically he informs us that in a right angled triangle such the one below the square of the hypotenuse is equal to the sum of the ıares of the other two sides.



i.e. $c^2 = a^2 + b^2$

This can be used in straight line geometry to work out the length of a line e.g.



Take length of line as L

then $L^2 = 3^2 + 4^2$

$= 25$

so $L = 5$

Try changing this instruction in the solution of exercise 1(d) to see an example of how Pythagoras can justify his existence:

```
50 IF X*X + Y*Y>1849 THEN GO TO 80
```

and a very nice set of equal length lines are produced in an arc.

Beware when using Pythagoras' theorem, particularly in loops, because the SQR function and even powers of numbers are very slow to evaluate. For example the following statement has the same effect as the instruction above but it is much slower:

```
50 IF SQR(X**2 + Y**2) > 43 THEN GO TO 80
```

Try it and see.

1.6 MORE STRAIGHT LINES

Lines Through a Point

Having done a quarter of a spiders web above, why not try a full web-shape. To do this we need to know some more theory about equations of lines on a graph, and in particular how to calculate the equation of a line between two points.

For example, say we want to draw a line between (2,3) and (15,20). Both of them are on the line (general equation $y = mx + c$) so both satisfy its equation.

So for point (2,3) we have $3 = 2m + c$
and for point (15,20) we have $20 = 15m + c$

and then we have another mathematical unpleasantry, a pair of simultaneous equations! We eventually find that in a general case, the equation through two points (p,q) and (r,s) is obtained by

$$\frac{y - q}{s - q} = \frac{x - p}{r - p}$$

Enough of the theory, let's draw some more pictures. We want to get a web or star shape, with the centre at the centre of the screen, (32,22). Therefore we want to draw lines from different points on the y axis through (32,22). This makes things easier since the y axis has the equation $x = Ø$.

So taking $(p,q) = (32,22)$
and $(r,s) = (Ø,y)$

we get $$\frac{y - 22}{s - 22} = \frac{x - 32}{-32}$$ for S from Ø to 43

which after a lot of bashing comes to

$$y = 22 - \frac{(x - 32)(s - 22)}{32}$$ for S from Ø to 43

giving the following program

```
1Ø FOR S=Ø TO 43 STEP 5 . . . (Use 39 rather than 43
2Ø FOR X=Ø TO 63                      with 1K ZX81's)
3Ø LET Y=INT(22–(X–32)*(S–22)/32)
4Ø IF Y>43 OR Y<Ø THEN GO TO 7Ø
5Ø PLOT X,Y
6Ø NEXT X
7Ø NEXT S
```

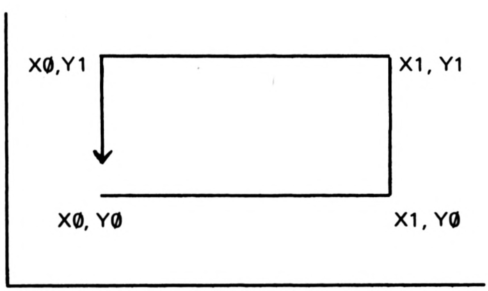
Lines with a Given Slope

Exercise 1(e): The display from the program above does not give a complete web effect because lines are only drawn from the y axis. Extend it by working out the equation of lines through a point *with a given gradient* and thus produce a complete web.

Spirals

An interesting display can be produced by drawing lines around the outside of the screen which gradually move into the centre in a 'rectangular spiral'. It is also quite an interesting exercise in logic.

Consider a general case where we are somewhere in the middle of the display:



We can label the corners of the current rectangle as shown above. Therefore we initially set the values of XØ, X1, YØ, Y1 to be at the edges of the screen and then gradually change them in the course of the program to produce the spiral. However we have to be very careful as to *where* in the program we modify these values.

This works very nicely:

```
 10 LET XØ=Ø    ...   15 )
 20 LET X1=63   ...   48 )  for 1K ZX81's
 30 LET YØ=Ø    ...   20 )
 40 LET Y1=43   ...   43 )
 50 FOR X=XØ TO X1
 60 PLOT X,YØ
 70 NEXT X
 80 FOR Y=YØ TO Y1
 90 PLOT X1,Y
100 NEXT Y
110 FOR X=X1 TO XØ STEP -1
120 PLOT X,Y1
130 NEXT X
140 LET X1=X1-1
150 LET YØ=YØ+1
160 LET Y1=Y1-1
170 FOR Y=Y1 TO YØ STEP -1
180 PLOT XØ,Y
190 NEXT Y
200 LET XØ=XØ+1
210 GO TO 5Ø
```

We need to stop the process somewhere so add

```
165 IF YØ>=Y1 THEN GO TO 5ØØ
```

and if you want to check that we stop at the right place add

```
5ØØ PRINT "Ø"
```

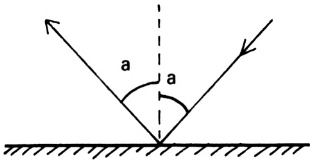
If you like this display and feel it could be extended to give a continuously moving video background in a room, you are absolutely right : wait for Section 1.7!

Bouncing

Many of the early TV games involved a ball bouncing around the screen.

We are now going to look at how to get a moving object to bounce off a flat object.

We assume that if our ball hits a wall at a certain angle it will bounce off at the same angle, i.e.:—



You are probably getting the sinking feeling that this is going to involve more theory: true, but not too much. It all has to do with the gradient of the line followed by the ball. We find that the gradient has its sign reversed after reflection from the wall. If you are into such things, this is because

$$\tan(90-a) = -\tan(90+a)$$

or incident gradient = −reflected gradient

Therefore if a ball travelling with a gradient of m hits a wall at a point (a,b) then it will continue with gradient negated and its equation will be

$$y = m(x-a) + b$$

or $y = mx + (b-ma)$

For drawing on the ZX81 we also have to be clear that if the ball hits a wall it will change direction on the screen, and therefore needs to be plotted carefully.

The following program draws a line starting at the origin on the screen at an angle specified by the user and then bounces it off the edges of the screen. Its path is left on the screen to illustrate the theory above. Use angles between 2Ø° and 8Ø° for useful results.

```
1Ø PRINT "ANGLE=";
2Ø INPUT A
3Ø CLS
4Ø LET M=TAN(A*PI/18Ø)
```

```
 50 LET X=0
 60 LET I=1
 70 LET C=0
 80 LET X=X+I
 90 LET Y=M*X+C
100 IF X<=0 OR Y<=0 THEN STOP
110 IF X>=43 OR Y>=43 THEN GO TO 140
120 PLOT X,Y
130 GO TO 80
140 LET M=-M
150 LET I=I-2*(X=43)
160 LET C=Y-M*X
170 GO TO 80
```

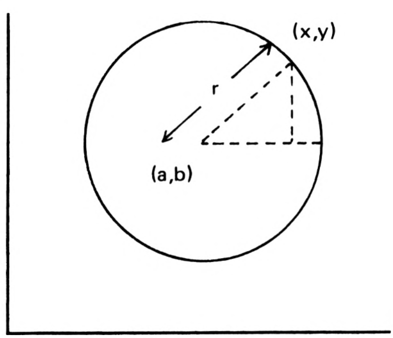
If you want to show just a single moving spot rather than continuous lines, add suitable UNPLOT and PAUSE statements.

1.7 CIRCLES AND OTHER INTERESTING SHAPES

We will use all this theory eventually in developing some good graphics games, so let us consider a final chunk of coordinate geometry.

Circles

We can specify the equation of a circle by noting that every point on the circle is the same distance away from the centre:



Taking the centre (a,b) and the radius as r then we can say for a general point (x,y) on the circle, using the ubiquitous pythagoras that

$$(x-a)^2 + (y-b)^2 = r^2$$

and we can take $x-a$ as $r \cos \theta$
and $y-b$ as $r \sin \theta$ where θ is any angle

since $(r \cos \theta)^2 + (r \sin \theta)^2 = r^2(\cos^2\theta + \sin^2\theta)$
$= r^2$ as it happens

Therefore we get $x = a + r \cos \theta$
and $y = a + r \sin \theta$

So let's see what the ZX81 makes of plotting a circle:

```
 1Ø PRINT "RADIUS=";
 2Ø INPUT R
 3Ø PRINT R;"bCENTRE:bX=";
 4Ø INPUT A
 5Ø PRINTA;"bY=";
 6Ø INPUT B
 7Ø PRINT B
 8Ø FOR Q=Ø TO 36Ø
 9Ø LET P=Q*PI/18Ø
1ØØ PLOT A+R*COS P,B+R*SIN P
11Ø NEXT Q
```

RUN the program and enter the radius followed by the x— and y— co-ordinates of the centre. Make sure that the circle does not go over the edge of the screen in any direction. Before your eyes a circle will appear, albeit slowly. The slowness results from the evaluation of cosines and sines at line 1ØØ — the ZX81 takes a long time to work these out.

Exercise 1(f): Work out the equation of a circle centred at the origin and radius 4Ø and therefore write a program to draw a circle quadrant (quarter arc) on the screen with radius 4Ø.

Finally we choose a selection of interesting shapes and show how they may be plotted.

Parabola

Here is a nice parabola

```
1Ø  FOR X=Ø TO 63
2Ø  LET Y=INT((2.52–0.04*X)*X)
3Ø  PLOT X,Y
4Ø  NEXT X
```

Ellipse

An ellipse is almost a general case of a circle or a parabola. Try this general ellipse plotter:

```
 1  PRINT "A=";
 2  INPUT A
 3  PRINT A;"bB=";
 4  INPUT B
 5  PRINT B
1Ø  FOR Q=Ø TO 36Ø
2Ø  LET P=Q*PI/18Ø
3Ø  PLOT A*(1+COS P),B*(1+SIN P)
4Ø  NEXT Q
```

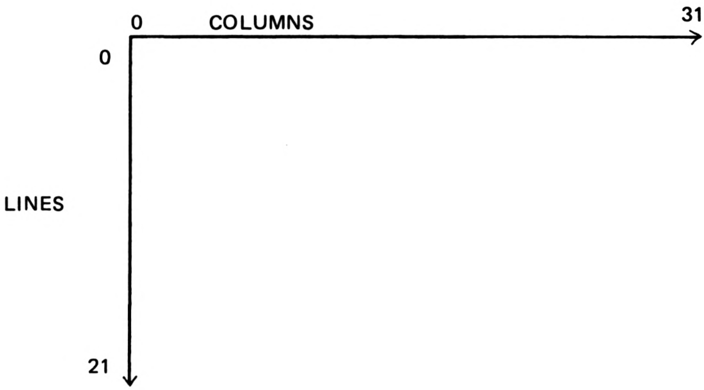
Try it with various values for A and B such as:

```
A = 30,   B = 20
A = 20,   B = 20
A =  5,   B = 21
```

1.8 DRAWING WITH OTHER CHARACTERS

The PRINT AT Instruction

All our graphics work so far has been of the 'join the dots' variety, since all the PLOT statement can do is to black-in squares. Fortunately this is not the limit of the ZX81's capability. The PRINT AT statement can also be used for picture drawing and it has one disadvantage but one considerable advantage over PLOT. The disadvantages is that it cannot address parts of the screen in so much detail as PLOT – the figure below shows its limitations



It can only draw in 22 x 32 positions and it works by means of specifying a line number and a column number, rather than standard x and y coordinates. Its great advantage is that any ZX81 character can be placed at a position.

The following simple routine illustrates the point

```
10 INPUT L
20 IF L<0 THEN STOP
30 INPUT C
40 INPUT S$
50 PRINT AT L,C;S$
60 GO TO 10
```

RUN the program and then keep entering groups of three items specifying line number, column number and character (or character sequence) and the ZX81 puts the character at this screen position.

ZX81 Video Show

Any of the programs previously considered can be modified to use PRINT AT rather than PLOT, and as promised here is a program to give a pleasant background video display to any room:

```
10 LET L0=0
```

```
 2Ø LET L1=21   ...  15 for 1K ZX81's
 3Ø LET CØ=Ø
 4Ø LET C1=31   ...  15 for 1K ZX81's
 45 LET Z$=CHR$(INT(RND*11+128*(RND<Ø.5)))
 5Ø FOR L=LØ TO L1
 6Ø PRINT AT L,CØ;Z$
 7Ø NEXT L
 8Ø FOR C=CØ TO C1
 9Ø PRINT AT L1,C;Z$
1ØØ NEXT C
11Ø FOR L=L1 TO LØ STEP-1
12Ø PRINT AT L,C1;Z$
13Ø NEXT L
14Ø LET L1=L1-1
15Ø LET CØ=CØ+1
16Ø LET C1=C1-1
17Ø FOR C=C1 TO CØ STEP -1
18Ø PRINT AT LØ,C;Z$
19Ø NEXT C
2ØØ LET LØ=LØ+1
2Ø5 IF LØ>=L1 THEN GO TO 5ØØ
21Ø GO TO 45
5ØØ CLS
51Ø RUN
```

It could even prove as addictive as 'Emmerdale Farm'!

or if you do not appreciate squares, how about circles?

```
1Ø FOR R=1Ø TO 2 STEP -1
15 LET Z$=CHR$(INT(RND*11+128*(RND<Ø.5)))
2Ø FOR Q=Ø TO 36Ø STEP 1Ø
3Ø LET P=Q*PI/18Ø
4Ø PRINT AT 1Ø+R*COS P,15+R*SIN P;Z$
5Ø NEXT Q
6Ø NEXT R
7Ø CLS
8Ø GO TO 1Ø
```

Try making your own variations — perhaps using a basic ellipse shape which grows fatter, thinner, longer or shorter with varying characters being used to draw it. You need not stick to graphics characters, many normal characters or inverse video characters can be very nice.

1.9 REALTIME

Instructions

At the beginning of the chapter we saw that realtime programs are ones in which the computer responds to user action **immediately**, no matter what other action it is currently taking. The ZX81 instruction that provides this facility is INKEY$. There is however another instruction which assists in a similar feature, moving displays, and that is PAUSE.

Almost all the programs in this section are designed for ZX81's running in compute and display mode (i.e. SLOW). However FAST mode is also available and in fact on 8K ROM ZX80's it is compulsory. In FAST mode, the only way to generate moving displays in BASIC is to cause the ZX81 to display the results of its processing by the PAUSE instruction, or rather PAUSE and POKE together since problems occur if you forget the accompanying POKE. We will put very little emphasis on PAUSE in this section, since INKEY$ is the centre of the ZX81's realtime facilities.

INKEY$

INKEY$ is at the same time the most peculiar and the most powerful instruction on the ZX81, and we hope that after reading this section and trying out the programs you will be rather better informed than if you only had access to Chapter 19 of the Sinclair Manual!

First of all let us get our terminology right. INKEY$ is not really an **instruction** like LET or IF but a **function**, since it is used as part of a ZX81 statement, and in fact has to be accessed via the FUNCTION key. Whenever the ZX81 executes a statement which includes INKEY$ it looks at the keyboard, and if a key is being pressed **at that instant**, the character of the key is put into INKEY$, i.e. if you were pressing 3 then

INKEY$ = "3"

If a key is not being pressed when the line containing INKEY$ is executed, then INKEY$ is set to the null string.

The following two line routine shows how it works

```
10 PRINT INKEY$;
20 GO TO 10
```

ꓤUN the program and then briefly touch any key on the keyboard. ₋et's assume you touched P — you will see a number of P's displayed on he screen. You may wonder why there are several rather than just one, ince you touched the key only once. To understand this you need to ıave an appreciation of how fast the ZX81 is computing (even in SLOW node!): while you have your finger on a key, albeit briefly, the ZX81 ycles round the GOTO 1Ø loop several times, the number of times ɔeing shown by the number of characters printed. Try pressing another ey, and as soon as you do you will see some more characters displayed ɔn the screen. See if you can touch a key so briefly that only one char-ıcter is displayed! While you are not touching a key, nothing is displayed ɔn the screen, since INKEY$ is the nullstring, and PRINT ""; produces ıothing. However if line 10 had read

```
1Ø PRINT INKEY$
```

.e. no semi-colon at the end, the program would have given quite a lifferent effect since PRINT "" causes a new line to be displayed, and he ZX81 quickly runs out of screen space.

'ry using the two line program above with entry of keys such as *, + ır =, ie. shift keys. You will see that depression of SHIFT has no effect, ɔut SHIFTed keys are displayed normally, even keywords. There are ıowever some exceptions eg: EDIT, FUNCTION, GRAPHICS and ꓤUBOUT. These all produce "?" on the screen, as does the NEWLINE key. Ve see from this that we can never enter graphic symbols via INKEY$ — ather a shame as we shall see later. Also SPACE is always interpreted s BREAK and this stops the program.

'o summarise where we have reached so far, we have seen that INKEY$ s a way of entering single characters into a program without the need for NPUT statements or even NEWLINE.

Moving Blobs in Realtime

'our reaction to the above treatment of INKEY$ may well be "OK, so vhat?" since it is not immediately obvious how INKEY$ can be used. ⅃opefully this little program may change your mind.

```
1Ø LET L = 10
2Ø LET C = 15
3Ø LET Z$ = "■" (inverse space)
4Ø IF INKEY$ = "5" THEN LET C=C−1
5Ø IF INKEY$ = "6" THEN LET L=L+1
```

```
 60 IF INKEY$ = "7" THEN LET L=L-1
 70 IF INKEY$ = "8" THEN LET C=C+1
200 PRINT AT L, C; Z$
210 GO TO 40
```

The program enables you to move a blob around the screen by pressing keys 5,6,7 or 8 and the blob moves according to the direction of the arrows on the keys. This is achieved by testing which key the user is pressing and changing accordingly the line and column numbers at which the blob is printed. RUN the program and see what interesting patterns can be produced. Only keys 5,6,7 or 8 will have any effect since these are the only values of INKEY$ for which the program takes any action. If the blob goes off a screen edge, the program generally crashes, so to overcome this add the following lines

```
80 LET L = L - L* (L=22) + 22* (L=-1)   ...  16 not 22 for 1K
90 LET C = C - C* (C=32) + 32* (C=-1)   ...  26 not 32 for 1K
```

and we get what is known as "wraparound" – if the line goes off one edge it reappears at the other edge.

It would also be pleasant if we had a choice of the type of character shown on the screen, rather than just a blob. With ZX81 technology all things are possible! Add this

```
100 LET K = CODE INKEY$
110 IF (K<>0 AND K<33) OR (K>36 AND K<64)
THEN LET Z$ = CHR$ K
```

If you now press any single character key other than 5,6,7 or 8 this character becomes the one being used for drawing on the screen. To stop any of these programs, simply press the SPACE key.

Note that in line 110 above we are careful to avoid taking a value of INKEY$ when no key is being pressed: we exclude it when it is equal to the null string (character code 0). Note also that line 110 has

```
LET Z$ = CHR$ K
```

rather than LET Z$ = INKEY$ as you might have expected. This is because it is possible that the value of INKEY$ might have changed between lines 100 and 110 (in particular it might be null) and this could cause inconsistencies in the program and therefore the resulting display.

1.10 EXAMPLE PROGRAMS

Introduction

In this section we will see how many of the concepts, and especially the maths, outlined above can be used in sophisticated realtime programs. Each program is given with detailed documentation so that the reader can understand how the program has been designed and developed. Appendix One shows the method of program design used and it is strongly recommended that a definite methodology should be used in programming. Although it is very tempting to start typing in BASIC instructions as soon as possible when developing a program, this causes more delay later, and it is in fact much quicker to design a program properly before touching the keyboard. Also, if a program is developed according to our method, documentation such as that given below builds up naturally so that you do not have to write it all up afterwards.

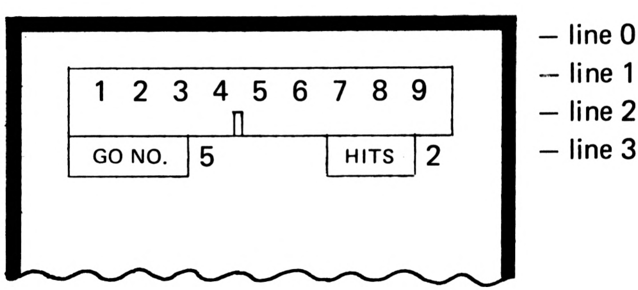
Anyway, on with the programs.

SHOOTING GALLERY (1K Memory)

Description

The program simulates a shooting gallery that you might find at a fair. An object moves from left to right across the screen under a row of numbers 1 to 9. The player attempts to hit the object by pressing one of the numeric keys 1 to 9 as the object passes under the number. There are ten goes and the program displays the current number of hits and the go number.

Sample Screen Format



Method

i. Set up screen and initialise hits H to zero
ii. Carry out the following with go number G = 1,2, ---, 10
   a. Set object position C to zero
   b. Clear shot line
   c. Clear message line & print go number
   d. Display object at position C
   e. If a key 1 – 9 pressed
      1. display shot
      2. if object hit:
         A. Display message
         B. Increment & display H
         C. Jump to (6)
      3. Wait for key to be released
      4. Increment C
      5. If C less then 31 jump to (d)
      6. Wait for 5 seconds
iii. Display end message & stop

List of Variables

H = no. of hits scored
G = go number (between 1 and 10)
N = number of key pressed (valid only for keys 1 – 9)
C = position of object on line

Program Listing

```
10   LET H=Ø
20   PRINT
30   PRINT "bbb1bb2bb3bb4bb5bb6bb7bb8bb9bbb"
     (inverse spaces & digits)
40   PRINT AT 4,Ø;"GO b NO." (inverse)
50   PRINT AT 4,26;"HITS" (inverse)
60   FOR G=1 TO 1Ø
65   LET C=Ø
7Ø   PRINT AT 3,Ø;"██████████████████████████████"
     (inverse – 30 spaces)
1ØØ  PRINT AT 4,6;G;TAB 12;"bbbbbbb"
11Ø  PRINT AT 3,C;"█ ▌" (inverse space and graphics 5)
13Ø  LET N=CODE INKEY$–28
14Ø  IF N<1 OR N>9 THEN GO TO 22Ø
```

```
15Ø   PRINT AT 3,N*3;"*" (inverse asterisk)
16Ø   IF N*3<>C+1 THEN GO TO 210
17Ø   PRINT AT 4,12;"GOT b HIM" (inverse)
18Ø   LET H=H+1
19Ø   PRINT AT 4,3Ø ;H
2ØØ   GO TO 24Ø
21Ø   IF INKEY$<>" " THEN GO TO 21Ø
22Ø   LET C=C+1
23Ø   IF C<>31 THEN GO TO 11Ø
24Ø   PAUSE 25Ø
25Ø   NEXT G
26Ø   PRINT AT 4,12;"THE b END" (inverse)
```

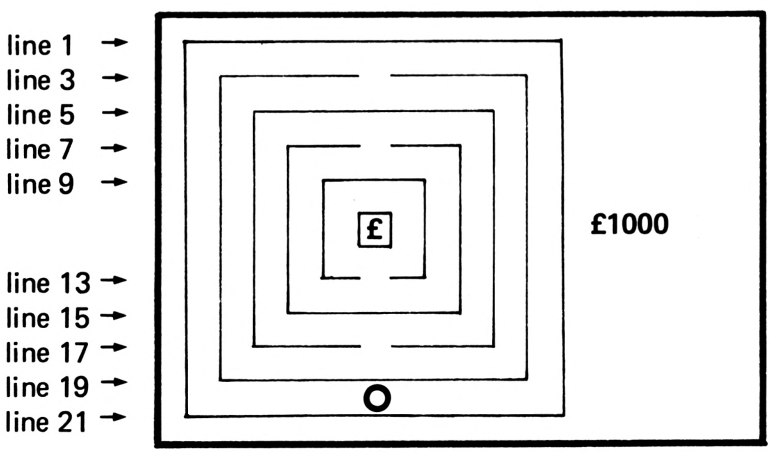
MONEY MAZE (16K Memory)

Description

A treasure chest full of £5 notes is located in the centre of a maze. You are on the outside of the maze and have to reach the treasure by using keys 5,6,7 or 8 to control your movement (direction arrows). However the chest has caught fire and the longer you take the less money there will be.

The program sets up and displays a 21 x 21 maze. The treasure chest is shown by £ and the player by O. A running counter of the amount of money left is shown to the right of the screen. The maze is displayed on the screen so that element i,j is at line i column j.

Sample Screen Format



Method

Array A of size 21 x 21 is used to hold the maze, with walls held as 128, space as Ø and the cash is 14Ø (inverse £).

- i. Set up array as shown
- ii. Print array in character form
- iii. Set sum of money M to 1000
- iv. Display M
- v. Set player's position at bottom of maze, line L and column C.
- vi. Display player's position
- vii. Pause to allow player time to see screen
- viii. Display player's position
- ix. Burn a fiver from M, and if M is zero, display message and stop
- x. Display M
- xi. Read number N from keyboard
- xii. If N between 5 and 8
  - a. Use N to update L and C to LI and CI
  - b. If position (CI, LI) is the chest print message and stop
  - c. If position (CI, LI) is space (not a wall)
    1. rubout position (C,L)
    2. change C to CI
    3. change L to LI
    4. jump to (viii)
- xiii. jump to (ix)

List of Variables

- A — array of size 21 x 21 holding maze
- I,J — loop counters used in setting up array
- M — amount of money left
- L — line no. of player's position
- C — column no. of player's position
- N — code number of key pressed (valid for 5 to 8)
- LI — new line no. of player's position
- CI — new column no. of player's position

Program Listing

```
10  DIM A (21,21)
20  FOR I=Ø TO 8 STEP 2
30  FOR J=I+1 TO 21-I
```

```
40  LET A(I+1,J)=128
50  LET A(21-I,J)=128
60  LET A(J,I+1)=128
70  LET A(J,21-I)=128
80  NEXT J
90  NEXT I
100 LET A(3,11)=0
110 LET A(7,11)=0
120 LET A(13,11)=0
130 LET A(17,11)=0
140 LET A(11,11)=140
200 PRINT
210 FOR I=1 TO 21
220 PRINT "b";
230 FOR J=1 TO 21
240 PRINT CHR$ A(I,J);
250 NEXT J
260 PRINT
270 NEXT I
300 LET M=1000
310 PRINT AT 11,22;"£";M
320 LET L=20
330 LET C=11
332 PRINT AT L,C;"O"
336 PAUSE 500
340 PRINT AT L,C; "O"
350 LET M=M-5
355 IF M<0 THEN GO TO 600
356 IF M<100  THEN PRINT AT 10,
    22;"HURRY"
360 PRINT AT 11,23;M;"bbb"
370 LET N=CODE INKEY$-28
380 IF N<5 OR N>8 THEN GO TO 350
390 LET LI=L-(N=7)+(N=6)
400 LET CI=C+(N=8)-(N=5)
410 IF A(LI,CI)=140  THEN GO TO 500
420 IF A(LI,CI)<>0 THEN GO TO 350
425 PRINT AT L,C;"b"
430 LET L=LI
440 LET C=CI
450 GO TO 340
500 PRINT AT 10,22;"YOU GOT"
```

```
510 STOP
600 PRINT AT 10,22;"TOO SLOW"
```

DUCK SHOOT (16K Memory)

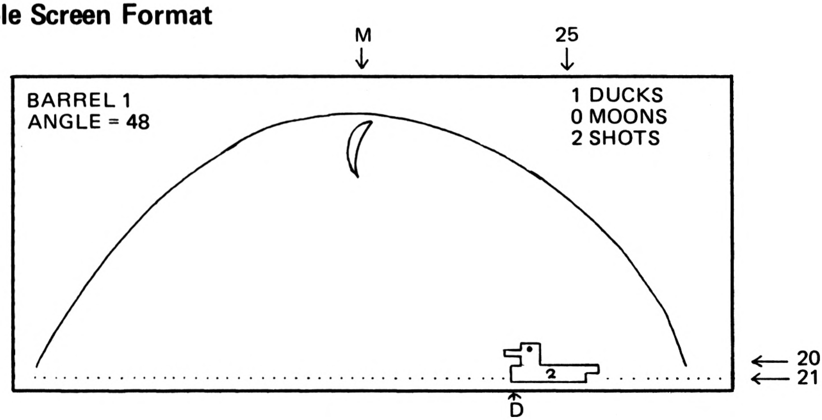
The author wishes to thank the designer of a similar game for the Research Machines 380 Z Microcomputer for the idea behind "Duck Shoot", the author's first experience of graphical games on a micro.

Description

A picture of a duck on a pond is displayed on the screen with the moon in the sky. The object is to shoot the duck making sure that you do not hit the moon in the process. Shooting is done by means of a double-barrelled cannon at the bottom left of the screen which fires up into the sky and the cannon ball travels in a parabola to eventually hit the pond, and hopefully the duck. The player chooses the angle of elevation of the cannon. There are five goes.

For each go the duck and the moon are displayed at different (random) positions. The duck is drawn at a position starting between columns 12 and 27 at the bottom of the screen, and the moon starting between columns 12 and 18 at the top of the screen. Scores are shown at the top right of the screen while the barrel number and angle are displayed at the top left. The number of the go is shown on the duck itself. The duck has a range of comments which it makes depending upon the accuracy or otherwise of the player's shot. The moon drops out of the sky if the cannon ball hits it.

Sample Screen Format



Method

(i) Initialise scores HD(ducks), HM(moons) and S(shots) to zero
(ii) Carry out the following for go number G = 1 . . . 5
  (a) clear screen
  (b) choose duck position D between 12 & 27 along line
  (c) choose moon position M between 12 & 18 along line
  (d) draw screen display and headings
  (e) carry out the following for barrel number B = 1 and 2
    (1) Display barrel number
    (2) Enter angle of elevation A
    (3) If angle not between 45° and 85° go to (2) above
    (4) Display angle
    (5) Plot path of cannonball. For each plot position (x,y)

      A. If (x,y) is on the moon
        (i) Increment HM (moon hits)
        (ii) Drop moon out of sky
        (iii) Increment and display S (shots)
        (iv) Go to (f) below

      B. If (x,y) is on the duck
        (i) If (x,y) is a central hit
          (a) Display "DEAD"
          (b) Increment HD (duck hits)
          (c) Increment and display S (shots)
          (d) Go to (f) below
        (ii) Display "OUCH"
        (iii) Go to (7) below
    (6) Display "MISS"
    (7) Increment and display S (shots)
    (8) Wait
  (f) Wait
(iii) Clear screen
(iv) Display final score of ducks hit.

List of Variables

HD = no. of ducks killed
HM = no. of times moon hit
S = no. of shots fired

G = go number (1–5)
D = starting x-axis position of duck
M = x-axis position of moon
B = barrel number (1 or 2)
A = angle of elevation of cannon (valid for 45° –85° only)
I = loop counter for display of cannon ball's path
X = x position of cannon ball
Y = y position of cannon ball
M2 = 2 times M
D2 = 2 times D
Z = loop counter for display of falling moon

Equation

The path of the cannon ball is plotted using the following equations:

X = INT (Ø.Ø14*1*(9Ø –A))
and Y = INT (I*(1ØØ–I)*Ø.Ø172) for I = Ø . . . 1ØØ

The X equation is chosen so that the cannon ball lands in the pond at the far right of the screen when angle A is 45°. The Y equation is chosen so that the cannon ball reaches its maximum height when I = 5Ø, i.e. in the middle of its flight path.

Program Listing

```
  2  LET HD = Ø
  4  LET HM = Ø
  6  LET S = Ø
 1Ø  FOR G=1 TO 5
 2Ø  CLS
 3Ø  LET D=INT(RND*16)+12
 4Ø  LET M= INT(RND*7)+12
 45  LET M2=M*2
 46  LET D2=D*2
 5Ø  PRINT "BARREL"
 6Ø  PRINT "ANGLE=?"
 7Ø  PRINT AT Ø,25;HD;"DUCKS"
 8Ø  PRINT AT 1,25;HM;"MOONS"
 9Ø  PRINT AT 2,25;S;"SHOTS"
1ØØ  PRINT AT 1,M;"▞"
11Ø  PRINT AT 2,M;"▌"
```

```
12Ø PRINT AT 3,M;"▚"
13Ø PRINT AT 21,Ø;" ................................"     (32 dots)
14Ø PRINT AT 2Ø,D;"–[;] bb/"
15Ø PRINT AT 21,D;"▐█";CHR$(G+156);"▛"
16Ø FOR B=1 TO 2
17Ø PRINT AT Ø,6;B
18Ø PRINT AT 1,6;"?b"
19Ø INPUT A
2ØØ IF A>85 OR A<45 THEN GO TO 19Ø
2Ø5 PRINT AT 1,6;A
21Ø FOR I=Ø TO 1ØØ
22Ø LET X=INT(Ø.Ø14*I*(9Ø–A))
23Ø LET Y=INT(I*(1ØØ–I)*Ø.Ø172)
24Ø PLOT X,Y
25Ø IF X<M2 OR X>M2+1 OR Y>41 OR Y<36 THEN GO
    TO 4ØØ
27Ø LET HM=HM+1
28Ø FOR Z=1 TO 18
29Ø PRINT AT Z,M;"b"
3ØØ PRINT AT Z+1,M;"▞"
31Ø PRINT AT Z+2,M;"▌"
32Ø PRINT AT Z+3,M;"▚"
33Ø NEXT Z
34Ø LET S=S+1
35Ø PRINT AT 2,25;S
36Ø GO TO 53Ø
4ØØ IF X<D2 OR X>D2+9 OR Y>3 THEN GO TO 495
41Ø IF X=D2 OR X=D2+1 OR X=D2+8 OR X=D2+9 THEN GO
    TO 45Ø
42Ø PRINT AT 2Ø,D–4;"DEAD"
43Ø LET HD=HD+1
432 LET S=S+1
434 PRINT AT 2,25;S
44Ø GO TO 53Ø
45Ø PRINT AT 2Ø,D–4;"OUCH"
491 GO TO 5ØØ
495 NEXT I
497 PRINT AT 2Ø,D–4;"MISS"
5ØØ LET S=S+1
5Ø5 PRINT AT 2,25;S
51Ø PAUSE 15Ø
52Ø NEXT B
```

```
530 PAUSE 150
540 NEXT G
550 CLS
560 PRINT AT 10,10;"THE END"(inverse)
570 PRINT AT 12,3;"YOU KILLEDb";HD;"bDUCKS"
```

Several more games are included in Chapter Three.

CHAPTER TWO – INFORMATION PROCESSING

2.1 INTRODUCTION

This chapter is aimed at readers who want to use a ZX81 with 16K RAM to store and retrieve quantities of information, i.e. who want the micro-computer to act as an electronic filing system. The objective may be to design programs to assist in leisure activities or in small businesses. If you do not have a 16K RAM pack you will not be able to use much of the material in this chapter, but perhaps as you read through you will be encouraged to invest in one!

Data and Data Processing

RG Anderson in his book 'Data Processing and Management Information Systems' defines **data processing** as "the systematic recording, arranging, filing, processing and dissemination of facts". The term is often used synonymously with business computing as against scientific or technical computing. As a general rule business data processing involves the simple manipulation of large quantities of information while technical computing involves the complex manipulation of small quantities of information.

For example, a typical data processing activity might involve stock control: here a large number of records are maintained but the most complex processing involved would be simple addition or subtraction for goods received or despatched respectively. Contrast this with a typical technical computing activity, the evaluation of sets of equations: here a small set of coefficients is used as data but complex matrix arithmetic has to be used to produce the solutions.

It is the author's opinion that data processing is a much more realistic function for a home computer than technical computing. Many people would like a computer to handle all their filing, from addresses and telephone numbers to recipes, but how many require trigonometric and logarithmic processing capabilities? The only possible application of such facilities is in games (certainly a useful way to use a home computer), but in general, sophisticated maths is not required. It is unfortunate that home computer manufacturers, Sinclair Research included, have addressed themselves more towards providing these technical computing facilities rather than data processing facilities such as large main memory capacities and good backing storage. In other words, Mr. Sinclair, why not forget about ARCSIN, ARCCOS and LN and give us a megabyte of online storage instead!

Hobbyhorses aside, a 16K ZX81 can be used for some very useful small tasks on the data processing side and we hope to give you the tools to develop your own such programs in this chapter.

2.2 CHARACTER HANDLING

Character Processing

While technical computing is mainly concerned with crunching numbers together, data processing deals largely with characters, either alphabetic characters or numbers not used for arithmetic purposes (e.g. code numbers). It is therefore essential that the reader has a good grasp of ZX81 character handling before embarking upon an information processing project. We suggest that you read over Chapters 7 and 21 of the Sinclair Manual and then follow the sections below.

Dimensions of Strings

A string can be used without first DIMensioning it, but giving a string a dimension can be useful if we always want it to be of fixed length. As an example, try this:

```
10 DIM A$(3)
20 PRINT "ENTER A WORD"
30 INPUT A$
40 PRINT A$
50 GO TO 20
```

The program will print the first three characters of any word you enter because A$ can only contain three characters.

To stop the above program is difficult, but possible: rubout the quotes around the cursor when invited to enter a word and then enter CHR$ (99**99). This causes the ZX81 to attempt to evaluate 99^{99} – a number too large for it to hold – so it crashes with error code 6.

Returning to dimensions of strings, it can be very useful to define a one character string which will always be used to INPUT responses to questions in a program, e.g.

```
 10 DIM Z$(1)
 . . .
100 PRINT "DO YOU WANT TO CONTINUE?"
110 INPUT Z$
120 IF Z$="N" THEN STOP
```

You may wonder why we bother with a dimension – why not do this:

```
100 PRINT "DO YOU WANT TO CONTINUE"
110 INPUT Z$
120 IF Z$(1)="N" THEN STOP
```

The answer is that if the user makes a null entry, i.e. just presses NEW-LINE, the first version is OK but the second version crashes with code 3 at line 120 because Z$(1) does not exist! It is vital in data processing programs which other people will use that the INPUTs be made as idiot-proof as possible. (See BOMB-PROOFING in Section 2.5).

Substrings

As we will see later in this chapter, verification of information input to a program is very important. Otherwise bad data gets onto files and it tends to make the whole system look ropy. A common verification is to check whether a string is alphabetic, i.e. contains letters A ... Z only.

```
 10 PRINT "ENTER ALPHABETIC WORD"
 20 INPUT A$
 30 IF A$=""THEN GO TO 20
 40 PRINT A$;
 50 FOR I=1 TO LEN A$
 60 IF A$(I)<"A" OR A$(I)>"Z" THEN GO TO 100
 70 NEXT I
 80 PRINT "IS ALPHABETIC"
 90 STOP
100 PRINT "HAS ERROR CHARACTER AT POSITION";I
110 PRINT "PLEASE RE-ENTER"
```

120 GO TO 20

Several techniques are employed in this program. Firstly at line 30 – always test for a null input and if one is found go back to the INPUT statement: try this out and see the effect from the user's end. Secondly at line 60 – relational operators work with characters as well as numbers. Lastly at line 100 – error messages should be as precise as possible to inform the user what he is doing wrong.

N.B. Be careful when using this routine because it treats 'space' as non-alphabetic, so "JOHN SMITH" would be rejected as non-alphabetic.

> **Exercise 2(a):** modify the above program to allow spaces and hyphens as well as letters A to Z.

We often need to determine whether a string contains a given word or sequence of characters.

> **Exercise 2(b):** write a program to enter a sentence and then test whether it contains the word "THE" and give an appropriate message.

2.3 DESIGN OF DATA PROCESSING PROGRAMS

Systems Analysis

In the world of business computing the analysis, design and implementation of computerised systems is a profession in itself. Obviously we are not going to call in a professional systems analyst to design programs for our 16K ZX81, but many of the methods used by the professionals can be be scaled down and applied for our purposes.

Is It Feasible?

One of the first stages in systems analysis is the feasibility study – a survey of whether the area under study can usefully be computerised. Many data processing tasks are best done manually rather than by computer, and this applies especially to home DP. For example you may have some excellent ideas for a Recipe Access and Testing System for your spouse but how feasible is it that he/she will use RATS on a day-to-day basis? Do you have enough extra sockets or even room in the kitchen for a ZX81, TV and cassette recorder? Will you be able to convince him/her that it is ten times better than his/her present manual system? Can the ZX81 cope with RATS storage requirements? Clearly these questions need honest answers before embarking upon a project which could consume many hours of precious time. A few of the areas you should consider are listed below:

BENEFITS	–	what substantial advantages would a computerised system have over the present system?
ZX81 CAPACITY	–	can the ZX81 store the program routines and data necessary for the system?
TECHNICAL ABILITY	–	are you sufficiently knowledgeable about the system and the relevant ZX81 facilities to implement your aims?
TIMESCALE	–	can the system be implemented in the time available?
USE	–	will the system be regularly and conscientiously used by the person(s) for whom it is designed?

Only after getting a positive answer to the above questions should you proceed with the design.

How Is It Done Now?

Before designing a new system a systems analyst takes a detailed look at how the present system operates using techniques such as interviewing staff, examination of documents, questionnaires and observation. If you are designing a system to be used by yourself you will have a clear idea of how you currently handle things and how things could be improved. However if you are producing a program to be used by someone else, you must get all this information from them. Since we are considering mainly filing systems on the ZX81 you need details of

Number of file records – present and future requirements
Size of records
How records are identified
How often records are added, changed or deleted.
Typical contents of records

How records are processed
Checking procedures

When we discuss the design of DP programs you will see why such facts are needed.

How Should It Be Done?

When getting together your ideas regarding features to be included in the computerised DP system, you need to be clear of the limitations of the current system and how these could be overcome. Eventually of course the facilities to be provided need to be listed in detail and a program routine designed to provide each facility. Appendix One describes a programming methodology that works: it is very important when writing a large program that a considerable amount of detailed program design is put in before touching the ZX81 keyboard.

You will need to pay particular attention to record formats and screen formats, i.e. what will be held in a file and what will appear on the screen. File formats are discussed in detail in Section 2.4. Good screen formats are vital for a workable program, particularly if the program will be used by a non-computer specialist, for example your husband or wife. Typically, in a section of the program to allow you to add new records to the file, the information presented on the screen should clearly and concisely describe what data needs to be entered and in what order. Features such as inverse video and judicious use of PRINT AT statements can make the program very user-friendly rather than user-nasty. It is best to actually map out on a piece of graph paper what will appear on the screen at major points in the program, and this can then be used to give you line and column numbers when you come to program your PRINT statements.

The Moment of Truth

Having designed the program, written it and debugged it according to the rules in Appendix One, the time comes to actually use it — "go live" in computing terminology. If someone else is using the program make sure that they are well-informed as to what to do to reap the amazing benefits offered by the program or else your efforts will have been wasted. In fact, if your family or others will be using your masterpiece of the programmer's art, a major exercise on your part will be selling the system to them and training them: people will not use a computer (particularly if they object to nasty electronic objects and trailing wires) unless they are convinced it will help them in their own tasks or activities.

A final word of caution — a "parallel implementation" is often the best way of introducing your new system. In other words do not burn all your address books and telephone directories on the day that you introduce your Computerised Address and Telephone System. There is the remote possibility that someone might try something that you had not thought of and a hitherto unnoticed bug in CATS will jump out and grab the ZX81 by the throat; or even the not unheard of vagaries of 16K RAM packs could make the system die just after you have typed in a hundred and fifty names and addresses.

2.4 DATA STRUCTURES

Definitions

In this section we consider how information may best be organised for use as an 'electronic filing system'.

First we define the terms used. A **field** is an item of data on a particular topic. A **record** is a collection of fields with some feature in common, and a **file** is a collection of related records, often organised in order.

To illustrate this terminology we introduce a sample application, a club membership list. This example will be used as the basis for all the concepts introduced in later sections of the chapter also. Assume that the list is currently kept by means of cards in a box, one card per member. The **file** is then the collection of cards in the box, while a **record** is an individual card and a **field** is some item on the card, e.g. name.

A card might look like this:

BETELGEUSE BREAKERS CLUB

Membership Form

Name ....Ford Prefect....

Address ....23 Chatham Gardens....

....London....

Post Code ....SW1X 1LB....

Telephone ....01 - 235 - 9649....

Special Interest ....Demolition....

Membership Number ....42....

Handle ....Earthman....

N.B. For the uninitiated, HANDLE is the code name used to identify a breaker or CB user

Files

The formats of records and fields are very important since the file forms the heart of the data processing system. Each heading on the card above will be a field on a member's record in the Betelgeuse Breakers Club (BBC) system which we are now starting to design. Whereas on a card we can have a few dotted lines upon which can be entered information, in a computerised system we must be much more precise as to the length of fields. The maximum number of characters allowed for each field must be chosen carefully. Every character will take up a byte of ZX81 memory so brevity is to be encouraged, although clarity must not suffer as a result.

Assume we choose the following:

NAME — length 15 characters
ADDRESS — length 50 characters
POSTCODE — length 8 characters
TELEPHONE NUMBER — maximum of 10 digits
SPECIAL INTEREST — length 20 characters
MEMBERSHIP NUMBER — 3 digits
HANDLE — length 15 characters

Two fields above are numeric. As the reader is no doubt aware, numbers can either be stored in the ZX81 as characters or digits, e.g.

```
      LET A$="123"        (characters)
   or LET A=123           (digits)
```

As far as memory requirements are concerned, a character string is stored in N+2 bytes where N is the number of characters, while a number is always stored in five bytes. Therefore if a number is more than three digits long it is more economic to use numeric format than character format. Another consideration is the usage to which the number is put: if the number will be used in any calculations it may be best to store it in numeric format since arithmetic cannot be carried out on characters, although of course VAL can be used to convert a number from character format to numeric format.

It may be helpful to the programmer if we split some of the fields down into subfields. For example, if we want to access membership records by surname then we could make NAME split into FORENAME and SURNAME, or perhaps INITIALS and SURNAME. Similarly, a separate field for TOWN could be useful: it all depends, as we discuss below, on how the file will be accessed.

Tables

The SPECIAL INTEREST field merits more detailed attention. It is quite likely that the interests of the Betelgeuse Breakers can be classified into main areas. To save space on the file we could then choose some code or code number to identify each of these special interests. For example:

Code No.	**Special Interest**
1	Demolition
2	Pangalactic Gargle Blasters
3	Sirius Cybernetics Corporation
4	Improbability Drive
5	Interplanetary DX
6	Vogon Poetry

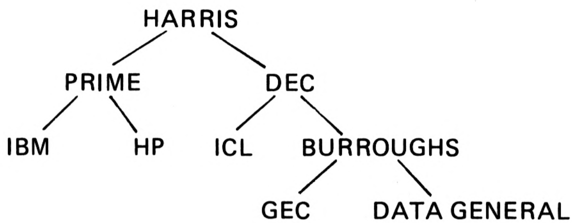
and so on.

If the actual interest had to be displayed somewhere in the program, a

table could be kept relating the code number to the special interest. In fact, for simplicity the code number could act as the subscript to a string array holding the special interests.

Other Data Structures

For completeness it should be mentioned that several other ways of organising data apart from simple files and tables are possible. Linked lists and tree structures can be very useful in certain applications. The diagram below shows a binary tree structure:



Such structures are implemented by means of each record having a left and right pointer to other records. Such organisation can be very useful in manipulating the data contained therein, since to move records around the tree (e.g. in sorting) involves only the resetting of pointers.

Files, Tables and the ZX81

Most microcomputers have facilities to store files of data on a secondary memory device such as cassettes or floppy discs. Unfortunately the Sinclair ZX81 does not. However, when a ZX81 **program** is SAVED on cassette the data used by the program (in variables and arrays) is stored as well, and it is this feature that enables us to consider file processing on a ZX81.

Returning to our example, BBC could use an array for each field on a record

e.g.	array N$	for NAMES
	array A$	for ADDRESSES
	array P$	for POSTCODES
	array T	for TELEPHONE NUMBERS
	array S	for SPECIAL INTEREST code numbers

array M	for MEMBERSHIP numbers
array H$	for HANDLE

Then a complete record would consist of a combination of members of these arrays, e.g. the first member would have his name stored in N$(1) address in A$(1), postcode in P$(1), telephone number in T(1) and so on. Assuming the system is designed for one hundred members, the arrays would be declared as follows:

```
DIM N$(1ØØ,15)
DIM A$(1ØØ,5Ø)
DIM P$(1ØØ,8)
DIM T(1ØØ)
DIM S(1ØØ)
DIM M(1ØØ)
DIM H$(1ØØ,15)
```

We can immediately calculate how much storage time this will occupy

array	N$	takes up about	100 x 15	=	1500 bytes
array	A$	" " "	100 x 50	=	5000 bytes
"	P$	" " "	100 x 8	=	800 bytes
"	T	" " "	100 x 5	=	500 bytes
"	S	" " "	100 x 5	=	500 bytes
"	M	" " "	100 x 5	=	500 bytes
"	H$	" " "	100 x 15	=	1500 bytes
				Total	10300 bytes

Another approach to file storage on the ZX81 is to have a single array, say R$, holding the records so that R$(1) = first record, and so on. In this case each field starts at a given position and the numbers must be stored in character form. We may describe the records in R$ using a RECORD FORMAT document such as the one following :–

BBC System	RECORD FORMAT	Member File
No. of Records = 100		Length of Record = 102 bytes
Name of Array = R$		

FIELD No.	FIELD NAME	START BYTE	END BYTE	LENGTH	VALUES
1	Name	1	15	15	
2	Address	16	65	50	
2·1	Street	16	45	30	
2·2	Town	46	65	20	
3	Postcode	66	73	8	
4	Tel. No.	74	83	10	Numeric
5	Interest Code	84	84	1	1-9
6	Membership No.	85	87	3	Numeric
7	Handle	88	102	15	

For 1ØØ records we declare R$ as DIM R$(1ØØ, 1Ø2) which will take up 1Ø2ØØ bytes approximately.

Notice that we have split ADDRESS into STREET and TOWN, that is the TOWN will always start at the 46th character position.

This is in fact the record format that we will use in this chapter to develop BBC programs. However it may help to mention an alternative in record design — that of **variable length fields.** In the record format above, much space will be wasted by data not filling their allowed field sizes. For example our sample record would be stored as :

```
1                 16                                  46
↓                 ↓                                   ↓
FORDbPREFECTbbb23bCHATHAMbGARDENSbbbbbbbbbbbbLONDON

                 66        74         84 85 88          102
                 ↓         ↓          ↓  ↓  ↓            ↓
bbbbbbbbbbbbbbSW1Xb8LBØ12359649b 1  42b EARTHMANbbbbbbb   (b=space)
```

Which contains a lot of unused space. With variable length working we store a special **terminator** symbol after each field, and it is this that indicates to the program the end of one field and the start of another. We could use inverse characters as terminators, e.g.:

FORDbPREFECT $\boxed{S}$23bCHATHAMbGARDENS$\boxed{T}$LONDON $\boxed{P}$SW1Xb8LB
$\boxed{T}$Ø12359649$\boxed{I}$1 $\boxed{M}$42 $\boxed{H}$EARTHMAN

which only takes up 71 bytes rather than 1Ø2 above. If this is an average saving of space, then with 1ØØ records we will save about 31ØØ bytes. The trade-off is that extra processing is required by the program to find and pick out specific fields. If you are short on file storage space this is certainly the technique to use, if your computer has the facilities to do this. Unfortunately the ZX81 does not since we are limited by the way in which the ZX81 handles string arrays. If we want to store 1ØØ records in a string array R$, we must dimension R$ thus
thus

DIM R$(1ØØ,N)

where N is the length of each record. Thus we must choose a fixed record length, although we may have variable length fields within a record.

Variable Length Records

The only way of implementing true variable length records is to store the entire file as one long string with separator symbols between each of the records. We could for example use inverse asterisks, e.g.

R$

RECORD 1	*	RECORD 2	*	RECORD 3	*	RECORD 4	*			etc.

In this way each record only takes up the number of bytes that it needs. However what you win on the swings you lose on the roundabouts and efficient storage formats require extra processing to access and use them. One fairly easy way of accessing records stored in this format is to set up a pointer array P, in which P(i) shows the starting position of record i in R$. So to extract record 15 from the array and put it into X$ we have

LET X$ = R$(P(15) TO P(16)–1)

Using this method we can dispense with the separator symbols and store one record immediately after another.

As an example we can write a program which invites the user to enter ten names; store the names in a single string N$ and set up pointers in array P to indicate the starting position of each name. Then we invite the user to enter a record number (between 1 & 1Ø) and extract and print out the appropriate record.

The program is listed below:

```
 1Ø DIM N$(3ØØ)
 15 DIM P(11)
 2Ø LET C=1
 3Ø PRINT "PLEASE ENTER 1Ø NAMES"
 4Ø FOR I=1 TO 1Ø
 5Ø PRINT TAB 5;I;")b";
 6Ø INPUT X$
 7Ø IF X$=""THEN GO TO 6Ø
 8Ø LET P(I)=C
 9Ø LET L=LEN X$
 95 LET N$(C TO C+L–1)=X$
1ØØ LET C=C+L
11Ø PRINT X$
12Ø NEXT I
125 LET P(I)=C
13Ø PAUSE 2ØØ
135 POKE 16437,255
14Ø CLS
15Ø PRINT AT 1Ø,Ø;"ENTER RECORD NUMBER OR Ø TO STOP"
16Ø INPUT N
165 IF N=Ø THEN STOP
17Ø IF N<Ø OR N>1Ø THEN GO TO 16Ø
18Ø CLS
19Ø PRINT AT 1Ø,7;"RECORD NUMBER";N
2ØØ PRINT AT 12,14;"IS"
21Ø PRINT AT 14,(31+P(N)–P(N+1))/2;N$(P(N) TO P(N+1)–1)
22Ø GO TO 13Ø
```

Method:

(i) Set current position pointer C to 1

(ii) Carry out the following for entry number I=1...1Ø

(a) Enter name X$
(b) Store C at position I in pointer array P
(c) Calculate length L of X$
(d) Insert name X$ into array N$ between positions C and C+L−1
(e) Update C to next free position in array N$
(f) Print name X$

(iii) Store final value of C in P(11)
(iv) Wait for 4 seconds
(v) Clear screen
(vi) Enter record number N
(vii) If N=Ø stop
(viii) If N not between Ø and 1Ø then go back to (vi)
(ix) Clear screen
(x) Display record number N by accessing between positions P(N) and P(N+1)−1 in array N$
(xi) Go back to (iv)

List of Variable Names

array N$	=	holds the 1Ø names as a single string
array P	=	holds pointers to starting positions of names in N$
C	=	shows next free position in array N$
I	=	loop counter indicating sequence number of name being entered
X$	=	name as entered
L	=	length of X$
N	=	record number to be printed

Comments

The technique of adding records to a single string is very useful and can be applied to records having multiple fields, each of which can themselves be of variable length.

N.B. The weird looking algebra at line 21Ø in the column position is to make sure that the name is printed centrally on the screen, whatever the length. As we will see in the next section, clarity or even prettiness of output gives greater user-friendliness (Programs with Pleasant Personalities).

2.5 FILE PROCESSING

Introduction

Having considered the different ways in which information can be stored in memory we can look at typical ways of processing it. We will first of all look at our example of the Betelgeuse Breakers Club (BBC) membership list in more detail, delving into what processing facilities would be required; we then think of what features need to be incorporated in file processing systems generally; and finally we split file processing down into typical modules such as file creation, validation, sorting and update, and use the BBC example to illustrate each of these concepts.

Sample Requirement

In Section 2.4 a typical Betelgeuse Breakers Club membership card was shown, and the file format for a computerised system was also explained (see page 46).

In the design of large computer program suites it is common for a user department to write a report specifying what facilities are required — an OPERATIONAL REQUIREMENT. Although we are considering information processing on a much smaller scale, it is still necessary to list what features our program aims to provide, since for every major facility a section will need to be included in the program.

The Secretary of the Betelgeuse Breakers Club will probably be looking for facilities in a computer system similar to the requirements of any Club Secretary. Let us assume these are:

- (i) finding a record by name, membership number or handle
- (ii) getting a list of all members' names and handles
- (iii) getting a list of all members interested in a certain topic.
- (iv) finding out which membership subscriptions are due: if, as is likely, membership numbers are handed out chronologically, this effectively means listing members with numbers in a certain range.

In addition there are certain run-of-the-mill facilities that must be available including adding and removing records and so on. These facilities will be formalised and developed as program modules in the final subsection of 2.5.

Program Features

Before seeing how a typical file processing program is written it is a salutary exercise to consider certain elements of programming style and technique. We aim to design systems which will be **bomb-proof** (or idiot-proof), **user-friendly** and **garbage-free.** Such terms may mean as little to the reader at this stage as redneck radio to an Easter Bunny, but all will be made clear.

BOMB-PROOFING is the careful design of programs and particularly INPUT sections so that the program cannot be made to terminate abnormally, (i.e. crash or bomb out). This is particularly important if the program user is not the program author. To achieve good bomb-proofing the program designer has to develop a very low opinion of the abilities of the intended user (even if it is himself), and hence the synonymous term idiot-proofing. In other words, if a mistake can be made, assume the user will make it.

As a simple example, consider a part of a program in which a number between Ø and 999 must be entered, e.g. as a membership number. Bomb-proofing theory suggests that we should tell the user the valid range and also check his entry:

```
100 PRINT "ENTER MEMBERSHIP NO. (0–999)"
110 INPUT M
120 IF M<0 OR M>999 THEN GO TO 110
```

We find that causing the program to wait until the user enters a correct number is usually sufficient, but some designers prefer to add an extra message, e.g.

```
110 INPUT M
120 IF M>=0 AND M<=999 THEN GO TO 150
130 PRINT "OUT OF RANGE:REENTER"
140 GO TO 110
150 . . .
```

However with this system, if the user persists in entering rubbish the number of error messages printed will eventually fill the screen and thus crash the system.

Nevertheless the major fault in the discussion so far is that if the user makes a non-numeric entry, e.g. WHAT, then the system crashes. The only way to get round this is never to have straight *numeric* INPUT statements but always to use strings and then convert them to numbers if they are valid, e.g.

```
. . .
1ØØ PRINT "ENTER MEMBERSHIP NO.(Ø–999)"
11Ø INPUT M$
12Ø FOR I=1 TO LEN M$
13Ø IF M$(I)<"Ø" OR M$(I)>"9" THEN GO TO 11Ø
14Ø NEXT I
15Ø LET M=VAL M$
16Ø IF M>999 THEN GO TO 11Ø
```

If several numbers are required to be input in a program it is a good idea to write a general subroutine to carry out the string-to-numeric conversion.

As far as string inputs are concerned, the main idiotic action to beware of is the null input, i.e. where the user just hits the NEWLINE key. In the last example above LEN M$ at line 12Ø evaluates to zero so the FOR . . NEXT loop is stopped and line 150 cannot be executed because VAL of the null string is incorrect (error code C). Therefore every string input must be followed by a test for the null string. Here:

```
115 IF M$=""THEN GO TO 11Ø
```

However the reader should note that if the string has previously been DIMensioned then this test will not work. This is because the ZX81 sets a string to all spaces when it is DIMensioned. You can show this by the following simple program

```
1Ø  DIM M$(3)
2Ø  INPUT M$
3Ø  PRINT ".";M$;"."
```

Do a RUN 2Ø and enter just NEWLINE : M$ is null. However including the DIM statement by RUN and following the exactly similar procedure causes three spaces to be printed for M$.

USER-FRIENDLINESS was the second objective in program design. We have already referred to this and in fact idiot-proofing is part of being user-friendly or perhaps user-condescending. It consists of making the program as easy to use and human-like as possible. Prompts should be in plain English wherever possible and screen formats should look nice with clear headings, central placing and highlighted where appropriate.

As suggested in Section 2.3 the program designer may map out each screen display on a piece of graph paper and check whether the display meets these objectives.

GARBAGE-FREE was the third quality required, and this refers to the old computer adage "GIGO" or "garbage in, garbage out". In other words if you accept incorrect data into a computer system then you will get incorrect results. Here we are not considering errors which cause a program to crash but rather errors which produce wrong results. This is particularly relevant when information is being fed in to be used as file records: once information is on file it may be difficult and messy to remove it.

Books on systems analysis theory list a vast range of checks which can be carried out on input data to avoid garbage being accepted onto file. Such validation methods include format checks, range checks, period checks, compatibility checks and many more. The alphabetic validation listed on page 37 is a typical example of making sure that numeric or special characters are not accepted into straight alphabetic fields such as that for a person's name.

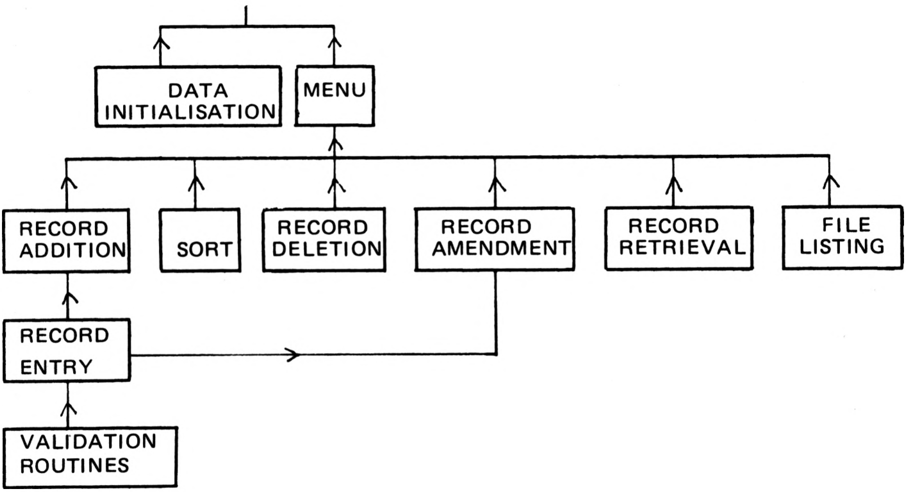
Validation techniques often include an element of redundancy and the use of a **check digit** is a good example. A check digit is an extra digit at the end of a code number which is formed by some specified calculation on the code number. The check digit would initially be calculated when the code-number was first allocated and then when the code is entered into a program, the program includes a routine to recalculate the check digit : if a discrepancy appears then the entry is incorrect.

Unfortunately the reader may find that in all these three aims, some limitations have to be made purely because of lack of ZX81 memory space.

Program Modules

All of the program modules required by the Betelgeuse Breakers Club will be included in a single ZX81 program. The program will be **menu-driven,** that is, a menu of options available will be displayed at the beginning, and after the option chosen is completed the program returns to the menu.

The modular structure of the program is shown below



DATA INITIALISATION

In the first section of the program we declare the arrays required and other initial data values.

The arrays used are:

N$	=	Member's name, length 15 characters
S$	=	Member's street, length 30 characters
T$	=	Member's town, length 20 characters
P$	=	Member's postcode, length 8 characters
B$	=	Member's telephone number, length 10 characters
C$	=	Member's interest code, length 1 character
M$	=	Member's number, length 3 characters
H$	=	Member's handle, length 15 characters
R$	=	the membership file, 80 records of 102 characters

Z$ = general user's response, length 1 character
I$ = table of interests, 9 records of 15 characters each
X$ = record number entry, length 3 characters
W$ = working space for sorting, length 1Ø2 characters

We also define N, the number of records currently on file. Note we are restricting this to a maximum of EIGHTY because of memory limitations.

This section is only used in the first program run to give initial values to data – All subsequent program runs are started by GO TO 5Ø so that previously defined data are retained.

```
1Ø  DIM N$(15)
12  DIM S$(3Ø)
14  DIM T$(2Ø)
16  DIM P$(8)
18  DIM B$(1Ø)
2Ø  DIM C$(1)
22  DIM M$(3)
24  DIM H$(15)
26  DIM R$(8Ø,1Ø2)
28  DIM Z$(1)
3Ø  DIM I$(9,15)
32  DIM X$(3)
34  DIM W$(1Ø2)
4Ø  LET N=Ø
41  LET I$(1)="Interest 1"
42  LET I$(2)="Interest 2"
43  LET I$(3)="Interest 3"
44  LET I$(4)="Interest 4"     . . . actual data chosen as required
45  LET I$(5)="Interest 5"
46  LET I$(6)="Interest 6"
47  LET I$(7)="Interest 7"
48  LET I$(8)="Interest 8"
49  LET I$(9)="Interest 9"
```

MENU:

The menu section displays the choice of options available and directs program control to the appropriate module (b=space, below).

```
 50  CLS
 60  PRINT TAB 4;"BETELGEUSEbBREAKERSbCLUB"
 70  PRINT
 80  PRINT "1.bADDbAbRECORD"
 90  PRINT "2.bSORTbRECORDS"
100  PRINT "3.bDELETEbAbRECORD"
110  PRINT "4.bCHANGEbAbRECORD"
120  PRINT "5.bDISPLAYbAbRECORD"
130  PRINT "6.bLISTbTHEbFILE"
140  PRINT AT 19,Ø;"ENTERbNO.bREQUIREDbORbØbTObSTC
150  INPUT Z$
155  IF Z$="Ø" THEN STOP
160  LET MO=CODE Z$-28
170  IF MO<1 OR MO>6 THEN GO TO 15Ø
180  GO TO 5ØØ *MO
```

MO is the menu option number chosen.

RECORD ENTRY:

Both option number 1 and option number 4 will require the entry of a record. In the former case, a new record will be added, whereas in the latter case an already existing record will be changed. However the entries will both require formatting and checking of inputs, so we write a general-purpose record entry module which can be used by both options.

The module will be entered with RN set to the number of the record to be entered. It then follows this method:

(i) Clear screen.
(ii) Display sequence number RN of record to be entered
(iii) Enter a member record:
  (a) Enter name into N$
  (b) Enter street into S$
  (c) Enter town into T$
  (d) Enter postcode into P$
  (e) Enter telephone number into B$, checking it is numeric
  (f) Enter interest code into C$, checking it is 1–9
  (g) Enter membership number into M$, checking it is Ø–999
  (h) Enter handle into H$
(iv) Display record as entered.
(v) User confirms or cancels record

(a) If confirmed (1) move N$, S$, T$, P$, B$, C$, M$ and H$ to R$ (RN)
(2) add 1 to RN
(3) Output confirmation message
(4) Go to (vi) below
(b) If cancelled (1) Clear screen
(2) Output cancellation message.

(vi) Invite entry of NEWLINE or M
(a) If NEWLINE, go to (i) above
(b) If M return to menu.

Notice that we do not put the user's entries straight onto the file R$: we demand positive confirmation of his entries before this happens (step (v) above).

The BASIC for this section appears below:

```
5ØØØ    CLS
5Ø1Ø    PRINT TAB 4;"ENTRYbOFbRECORDbNUMBERb";RN
5Ø2Ø    PRINT AT 2,Ø;"NAME:"; (inverse)
5Ø3Ø    INPUT N$
5Ø5Ø    PRINT N$
5Ø6Ø    PRINT AT 4,Ø;"ADDRESS" (inverse)
5Ø7Ø    PRINT AT 5,4;"STREET:"; (inverse)
5Ø8Ø    INPUT S$
51ØØ    PRINT S$
511Ø    PRINT TAB 4;"TOWN:"; (inverse)
512Ø    INPUT T$
514Ø    PRINT T$
515Ø    PRINT "POSTCODE:"; (inverse)
516Ø    INPUT P$
518Ø    PRINT P$
519Ø    PRINT AT 9,Ø;"TELbNO:"; (inverse)
52ØØ    INPUT B$
522Ø    GO SUB 9ØØØ
523Ø    IF NOT OK THEN GO TO 52ØØ
524Ø    PRINT B$
525Ø    PRINT AT 11,Ø;"INTERESTbCODE:"; (inverse)
526Ø    INPUT C$
528Ø    GO SUB 91ØØ
529Ø    IF NOT OK THEN GO TO 526Ø
53ØØ    PRINT C$
```

```
5310  PRINT AT 13,0;"MEMBERSHIPbNO:";(inverse)
5320  INPUT M$
5340  GO SUB 9200
5350  IF NOT OK THEN GO TO 5320
5360  PRINT M$
5370  PRINT AT 15,0;"HANDLE:"; (inverse)
5380  INPUT H$
5400  PRINT H$
5410  PRINT AT 19,0;"ISbTHISbCORRECT?"
5420  INPUT Z$
5430  IF Z$="N" THEN GO TO 5570
5440  IF Z$="Y" THEN GO TO 5460
5450  GO TO 5420
5460  LET R$(RN,1 TO 15)=N$
5470  LET R$(RN,16 TO 45)=S$
5480  LET R$(RN,46 TO 65)=T$
5490  LET R$(RN,66 TO 73)=P$
5500  LET R$(RN,74 TO 83)=B$
5510  LET R$(RN,84)=C$
5520  LET R$(RN,85 TO 87)=M$
5530  LET R$(RN,88 TO 102)=H$
5540  PRINT AT 19,0;"RECORDbADDEDbTObFILE"
5550  LET RN=RN+1
5560  RETURN
5570  CLS
5580  PRINT AT 19,0;"ENTRYbCANCELLED"
5590  RETURN
```

As you can see it is all good solid boring stuff — the meat of data processing. However, having it as a subroutine at least means we do not need to enter it twice.

The routine returns with RN incremented by one if a record has been entered onto the file, or the same if no record has been entered.

The screen format used is:

```
                    ENTRY OF RECORD NUMBER –

NAME: ____________

ADDRESS

        STREET: ______________________________

        TOWN: ________________

POSTCODE: ________________

TEL. NO: ___________

INTEREST CODE: __

MEMBERSHIP NO: ____

HANDLE: ____________

Message Line
```

VALIDATION ROUTINES

The record entry section invokes three subroutines to check the entry of telephone number, interest code and membership number. Each of the routines return a value OK, set to zero if the entry was invalid or one if it was valid.

Telephone number validation:

```
9000    LET OK=0
9010    FOR I=1 TO 10
9020    IF B$(I)="b" OR (B$(I)>="0" AND B$(I)<="9") THEN
          GO TO 9040
9030    RETURN
9040    NEXT I
9050    LET OK=1
9060    RETURN
```

Interest code validation:

```
9100    LET OK=0
9110    IF C$="b" THEN LET C$="0"
9120    IF C$<"0" OR C$>"9" THEN RETURN
9130    LET OK=1
9140    RETURN
```

Membership number validation:

```
9200    LET OK=0
9210    FOR I=1 TO 3
9230    IF M$(I)<"0" OR M$(I)>"9" THEN RETURN
9240    NEXT I
9250    LET M=VAL M$
9260    IF M=0 THEN RETURN
9270    LET OK=1
9280    RETURN
```

We can also have record number validation:

```
9300    LET OK=0
9310    IF X$(1)="b" THEN RETURN
9320    FOR I=1 TO 3
9330    IF X$(I)="b" OR (X$(I)<="9" AND X$(I)>="0")
          THEN GO TO 9350
9340    RETURN
9350    NEXT I
9360    LET VN=VAL X$
9370    IF VN>N OR VN=0 THEN RETURN
9380    LET OK=1
9390    RETURN
```

RECORD ADDITION:

This just adds a record at the end of the file and optionally repeats

```
500     LET RN=N+1
510     GO SUB 5000
520     IF RN=N+2 THEN LET N=N+1
530     PRINT AT 20,0;"PRESSbNEWLINEbTObADDbRECOR
          b";RN
540     PRINT AT 21,6;"ORbMbFORbMENU"
550     INPUT Z$
560     IF Z$="b" THEN GO TO 510
570     IF Z$="M" THEN GO TO 50
580     GO TO 550
```

RECORD AMENDMENT

This is very similar to record addition, but allows the user to re-input and therefore change a record already on file:

```
2ØØØ   CLS
2Ø1Ø   PRINT AT 2Ø,Ø;"ENTERbNUMBERbOFbRECORD"
2Ø2Ø   INPUT X$
2Ø3Ø   GO SUB 93ØØ
2Ø4Ø   IF NOT OK THEN GO TO 2Ø2Ø
2Ø5Ø   LET RN=VN
21ØØ   GO SUB 5ØØØ
22ØØ   PRINT AT 2Ø,Ø;"PRESSbNEWLINEbTObCHANGEb
         ANOTHER"
221Ø   PRINT AT 21,6;"ORbMbFORbMENU"
222Ø   INPUT Z$
223Ø   IF Z$="b" THEN GO TO 2ØØØ
224Ø   IF Z$="M" THEN GO TO 5Ø
2250   GO TO 222Ø
```

SORT:

It is likely that the membership file will be in membership number order, since as previously mentioned, numbers will probably be allocated in chronological sequence. However a sorting routine is included to allow for any anomalies.

There are many different methods of sorting information into sequence and such methods are easily found in computing textbooks. The following routine uses a simple exchange sort:

For pointer q from 1 to p :
If no. of record q > no. of record q + 1 then swop record q and q + 1

The routine uses characters 85 to 87 of record R$, the membership number: if sequencing is required on name or another attribute this specification may easily be changed.

```
1ØØØ   CLS
1ØØ2   PRINT AT 2Ø,Ø;"SORTING ..."
1ØØ6   FOR P=N−1 TO 1 STEP−1
```

```
1010    FOR Q= 1 TO P
1020    IF R$(Q,85 TO 87)<=R$(Q+1,85 TO 87) THEN GO
        TO 1060
1030    LET W$=R$(Q)
1040    LET R$(Q)=R$(Q+1)
1050    LET R$(Q+1)=W$
1060    NEXT Q
1070    NEXT P
1080    PRINT AT 20,0;"SORTbCOMPLETED"
1090    PAUSE 250
1100    POKE 16437,255
1110    GO TO 50
```

RECORD DELETION:

If someone leaves the Betelgeuse Breakers Club then their record must be deleted from file. To do this the user selects this option and specifies the sequence number of the record to be removed.

```
1500    CLS
1505    PRINT AT 20,0;"ENTERbNO.bOFbRECORDbFORb
        DELETION"
1510    INPUT X$
1520    GO SUB 9300
1522    IF NOT OK THEN GO TO 1510
1524    LET D=VN
1530    PRINT AT 21,0;"DELETING ..."
1540    FOR I=D TO N–1
1550    LET R$(I)=R$(I+1)
1560    NEXT I
1570    LET N=N–1
1580    CLS
1590    PRINT AT 19,0;"RECORDb";D;"bHASbBEENb
        DELETED"
1600    PRINT AT 20,0;"PRESSbNEWLINEbFORbMOREb
        DELETIONS"
1610    PRINT AT 21,6;"OR M FOR MENU"
1620    INPUT Z$
1630    IF Z$="b" THEN GO TO 1500
1640    IF Z$="M" THEN GO TO 50
1650    GO TO 1620
```

RECORD RETRIEVAL

One of the requirements of the Breakers Club Secretary was to retrieve a record by name, membership number or handle. This is how it is done:

```
2500  CLS
2510  PRINT AT 10,0;"SPECIFYbONEbOFbTHEb
      FOLLOWING:"
2520  PRINT AT 12,0;"NAME:"
2530  PRINT "MEMBERSHIPbNO:"
2540  PRINT "HANDLE:"
2550  PRINT AT 20,0;"ENTERbVALUEbORbNEWLINEb
      FORbEACH"
2560  INPUT N$
2570  PRINT AT 12,5;N$
2580  INPUT M$
2590  PRINT AT 13,14;M$
2600  INPUT H$
2610  PRINT AT 14,7;H$
2620  IF N$(1)="b"THEN LET N$(1)="*"
2630  IF M$(1)="b" THEN LET M$(1)="*"
2640  IF H$(1)="b" THEN LET H$(1)="*"
2650  PRINT AT 20,0;"SEARCHINGbbbbbbbbbbbbbbbbbbbbb"
2660  FOR I=1 TO N
2670  IF R$(I, 1 TO 15)=N$ OR R$(I,85 TO 87)=M$ OR
      R$(I,88 TO 102)=H$ THEN GO TO 2715
2680  NEXT I
2690  CLS
2700  PRINT AT 10,5;"NObRECORDbFOUND"
2710  GO TO 2810
2715  CLS
2720  PRINT TAB 10;"RECORDbNO.b";I
2730  PRINT AT 2,0;R$(I,1 TO 15)
2740  PRINT R$(I,16 TO 45)
2750  PRINT R$(I,46 TO 65)
2760  PRINT R$(I,66 TO 73)
2770  PRINT R$(I,74 TO 83)
2780  PRINT R$(I,84)
2790  PRINT R$(I,85 TO 87)
2800  PRINT R$(I,88 TO 102)
2810  PRINT AT 20,0;"PRESSbNEWLINEbFORbANOTHERb
      RECORD"
2820  PRINT AT 21,6;"ORbMbFORbMENU"
```

```
2830    INPUT Z$
2840    IF Z$="b" THEN GO TO 2500
2850    IF Z$="M" THEN GO TO 50
2860    GO TO 2830
```

Notice that it is helpful to the user to display a message to show that the ZX81 is busy doing something, e.g. SEARCHING at line 2650.

FILE LISTING:

This section shows how to implement a full or selective file listing. In the listing, only members' names and handles are displayed.

Options are:

(i) full listing
(ii) listing of members with a given interest
(iii) listing of members with membership numbers above a certain figure

```
3000    CLS
3010    PRINT TAB 10;"LISTbRECORDS"
3020    PRINT AT 20,0;"DObYOUbWANTbAbFULLbLISTb
        (Y/N)?"
3030    INPUT Z$
3040    IF Z$="Y" THEN GO TO 3065
3050    IF Z$="N" THEN GO TO 3120
3060    GO TO 3030
3065    CLS
3070    FOR I=1 TO N
3080    SCROLL
3090    PRINT AT 15,0;R$(I,1 TO 15);" ";R$(I,88 TO 102)
3100    NEXT I
3110    GO TO 3350
3120    PRINT AT 20,0;"SELECTbBYbINTERESTbORb
        NUMBER?b"
3130    INPUT Z$
3140    IF Z$="I" THEN GO TO 3170
3150    IF Z$="N" THEN GO TO 3260
3160    GO TO 3130
3170    PRINT AT 2,10;"INTERESTS"
3180    FOR I=1 TO 9
```

```
3190   PRINT I;".b";I$(I)
3200   NEXT I
3210   PRINT AT 20,0;"ENTERbINTERESTbCODEb
       NUMBERb(1–9)"
3220   INPUT C$
3230   GO SUB 9100
3240   IF NOT OK THEN GO TO 3220
3245   LET M$="999"
3250   GO TO 3300
3260   PRINT AT 20,0;" LISTbMEMBERSbWITHbNUMBERSb
       >?bbb"
3270   INPUT M$
3280   GO SUB 9200
3290   IF NOT OK THEN GO TO 3270
3295   LET C$="*"
3300   CLS
3310   FOR I=1 TO N
3320   IF R$(I,84)<>C$ AND R$(I,85 TO 87)
       <=M$ THEN GO TO 3340
3330   SCROLL
3335   PRINT AT 15,0;R$(I,1 TO 15);":";
       R$(I,88 TO 102)
3340   NEXT I
3350   PRINT AT 20,0;"PRESSbNEWLINEbFORbMENU"
3360   PAUSE 5000
3370   POKE 16437,255
3380   GO TO 50
```

Summary

When the above routines have been entered, records can be added and the system used. Once data has been entered, it is vital that the system is always started by GO TO 5Ø, since RUN automatically clears data. Always SAVE the system on cassette whenever any file additions or modifications have been made.

Although your own file processing application may not be identical to the Betelgeuse Breakers Club system, the same principles apply and many of the routines in this chapter may be used directly.

CHAPTER THREE – EDUCATION

3.1 THE ZX81 AS AN EDUCATIONAL TOOL

Introduction

At the time of writing Sinclair Research is operating a special offer to UK schools whereby a complete 16K ZX81 system with printer can be obtained at half-price. The offer came about in response to a Government-funded scheme to install a microcomputer in every secondary school, by providing a 50% subsidy for the purchase of either an Acorn Atom or a Research Machines 380Z.

Even without such a scheme for the Sinclair ZX80 this microcomputer found a place in many educational institutions. Certainly the ZX81 will prove even more popular. School students themselves will start to find that it is within their budgets, or rather their parents'. The ZX81 has many facilities that could make it a useful educational resource, but the key facility in education is of course suitable software. In this chapter we consider various types of educational computing and the design of software to be used in the primary and secondary sectors.

Computer Studies

Many secondary schools use microcomputers largely for examination subjects such as CSE or GCE 'O' level Computer Studies or GCE 'A' level Computer Science. These subjects generally require students to carry out substantial programming with a number of documented programs being submitted as part of the course assessment.

Most schools equipped with microcomputers allow and encourage students to get involved with programming. Even many primary schools in the author's region are encouraging children to develop their own programs with assistance from teaching staff.

In order for a microcomputer to be suitable for the learning of computer programming by a range of school students and also for more complicated project work for external assessment, the machine must be very flexible. The ZX81 scores well in this area and has a number of facilities that make program entry and development much easier than on many similar micros. In particular the entry of keywords by single key depressions and the automatic syntax checking of lines has been found to aid beginners considerably. The ZX81's line editing capability is also unusually sophisticated for a machine of its size.

For more advanced programming the ZX81 has many powerful number-handling and text-handling facilities permitting a range of applications programs to be developed. Obviously the addition of 16K memory extensions are vital for any degree of sophistication, but with this a great deal of potential is available. Finally, the addition of a printer at around £50 (projected at the time of writing) makes the system suitable for project work where hard copy is vital : in fact the cheapness of the printer is a considerable advantage over other systems.

On the negative side, ZX81 report codes are clumsy in program development. It is a nuisance having to look up the meaning of codes in order to find out why a program is going wrong. The tiny keyboard with its multifunction keys, while being a novelty and definite aid for those not familiar with keyboards, is a considerable disadvantage when entering long and complicated programs.

General Subjects

Increasingly schools are buying computers for use as educational aids in general subjects, rather than in computer studies. This seems to be a more realistic and sensible approach for many secondary students since it is much easier to recognise the value of computer technology if a student is already aware of actual problems which he later discovers a computer can help to solve. Similarly for primary schoolchildren a background in using micros as *tools* similar to cassette recorders or projectors enables them to encounter computers in their later educational or working lives without having inbuilt prejudices against the technology involved.

In this area therefore we are analysing the validity of the ZX81 as a black-box (!) providing educational facilities. Obviously a crucial aspect here is the quality and relevance of the software used, and since the best educational software is written by subject teachers rather than by computer people, this chapter aims to aid the reader in producing good software. However we can make some general comments about ZX81 BASIC and its relevance in this area of use. Presentation of information is very important in many subjects, and the graphics features of the

ZX81 are helpful although of limited resolution; similarly the SLOW compute and display facility is very useful in displaying active processes on the ZX81 screen. The inverse character displays available are also good but it could be argued that the provision of lower case characters would have been more helpful: it is a considerable disadvantage at the primary level or in any language work to have to use upper case characters only.

Being able to interact with computer programs in realtime, using INKEY: can be an advantage in the educational area, and exercises in coordinatioı can easily be designed using a limited number of keys. In fact the touch-sensitive keyboard can easily be partitioned off by overlays, with the active keys being labelled with special symbols according to the applicati currently in use.

Categories of Educational Programs

Before embarking upon the study of specific programs let us consider what general categories of programs can be used in the primary and secondary spheres to aid the teaching of non-computing subjects: this chapter does not seek to develop programs for Computer Studies, or for use in educational administration (see Chapter Two for ideas on this).

Demonstration or simulation programs are those with little student involvement, either run by a teacher or without any more inputs. Such programs are of limited value but can be used to demonstrate the facilitie of a machine to an introductory group or at a more advanced level to demonstrate some process that is difficult to explain or model otherwise, eg: a dangerous chemical experiment might be shown by the reagents anc products being displayed on the screen.

Programmed learning programs depend upon a program taking on to a small extent the role of a teacher: typically a program might contain teaching material on a topic and the student would respond to certain questions posed by the program. His response would determine the next block of material displayed by the program. Since these programs often require the storage of a great deal of textual material to be effectiv the value of such programs on the ZX81 is limited.

Test or **quiz** programs are a simpler variant of the previous category. Here a bank of questions is held in a program and the student is given each question in turn (or perhaps randomly from a large section of questions) and has to reply by entering his answer on the keyboard. The computer then responds by assessing his answer as right or wrong, and if the latter a hint or further help may be given. At the end a score appears. If the quiz is substantial the program may be written so that the student may stop in the middle, SAVE everything on cassette and restart from where he left off at a later time: the ZX81 is one of the few microcomputers which makes this very easy.

Modelling programs simulate a real-life process so that the user can be involved in the process without the accompanying problems or equipment. For example a complete list of chemical compounds with their reactions to certain tests can be held on file and the student can then be presented by the program with an unnamed compound and can perform a series of tests on it until he can finally determine what it is – all without getting his white coat dirty.

Games programs often have sound educational value, and many programs at primary level can be written as games to achieve their goals. Even quite stuffy programs such as maths quizzes can be written with a game-like flavour eg: a game of snakes and ladders in which the player has to get a sum right before he can throw dice.

A selection of different types of programs appears in the remainder of the chapter. Many 'standard' educational programs can be picked up easily from text books, magazines or groups such as MUSE, and it is not the author's intention to re-reproduce such material. Instead programs have been chosen which particularly use the educational facilities of the ZX81.

3.2 EDUCATIONAL PROGRAMS

MATHS STEPPING STONES (16K)

Background

The first program in this section has been chosen to illustrate how an essentially simple and boring maths testing program can be made interesting and dynamic using realtime graphics.

Here is the simple and boring version:

```
 10  LET S=Ø
 20  FOR I=1 TO 1Ø
 30  LET A=2+INT(RND*8)
 40  LET B=2+INT(RND*8)
 50  LET C=A*B
 60  PRINT AT I,Ø;I;TAB 2;")b";A;"bXb?b=b";C;"b"
 70  INPUT G
 80  IF G=B THEN GO TO 11Ø
 90  PRINT AT I,15;"NO.bANSWERbISb";B
100  GO TO 12Ø
110  PRINT AT I,15;"CORRECT" (inverse)
115  LET S=S+1
120  PRINT AT I,8;B
130  NEXT I
140  PRINT
150  PRINT "YOUbGOTb";S;"bRIGHTbOUTbOFb1Ø"
```

There are 1Ø questions of the type 4 x ? = 32 and the data names used are

S = number of questions answered correctly
I = question number
A)
B) the question posed in the form A X B = C
C)
G = user's attempt

There is nothing novel about this type of program and personal computer magazines and books are full of such things. Let us now consider a program which aims to test exactly the same principles but which does it in a much more attractive way.

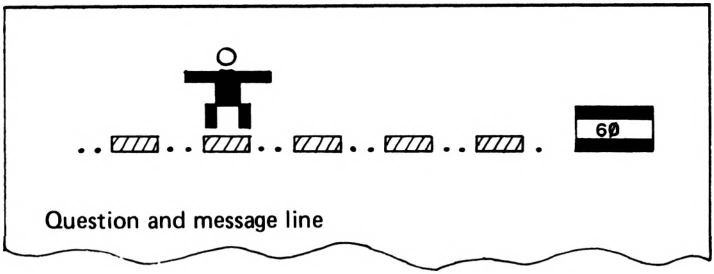
Description

The player is on one side of a river and a treasure chest is on the other side. The chest contains magic gold coins which as time goes on are turning into frogs: the longer the player takes to cross the river the more frogs and less gold coins. The player crosses the river by means of five stepping stones, but to reach each stepping stone he has to answer a maths question. If the player has not crossed the river after twenty questions the stones disappear and the player falls into the river. This also happens if all the coins have turned into frogs.

Method

- (i) Initialise score S to zero and coins CO to 1ØØ
- (ii) Draw river scene on screen
- (iii) For I from 1 to 2Ø
  - a) choose A at random between 2 and 9
  - b) choose B at random between 2 and 9
  - c) Evaluate C as A times B
  - d) Display I and question as A x ? = C
  - e) If a key has been pressed:
    - 1) If key = value of B then
      - (a) Display message
      - (b) Add 1 to S
      - (c) If S = 6 display message and stop
      - (d) Move man
      - (e) Go to (3) below
    - 2) Display message
    - 3) Pause
    - 4) Repeat to (iii)
  - f) Subtract 1 from CO and display CO.
  - g) Go to (e) if no key pressed above.
  - h) If CO is zero, go to (iv) below.
  - i) Repeat to (iii)
- (iv) Display message and stop.

Screen Format



Program Listing (16K)

```
 10 LET S=0
 20 LET CO=100
 30 PRINT AT 3,3;"..▒▒▒..▒▒▒..▒▒▒..▒▒▒..▒▒▒
    .";CO
 40 PRINT AT 2,29;"███"
 50 PRINT AT 4,29;"███"
 60 LET X=0
 70 GO SUB 500
 80 FOR I=1 TO 20
 90 LET A=2+INT(RND*8)
100 LET B=2+INT(RND*8)
110 LET C=A*B
120 PRINT AT 5,0;I;TAB 2;")b";A;"bxb?b=";C;
125 IF C<10 THEN PRINT "b"
130 PRINT AT 5,15;"bbbbbbbbbbbbbb"
140 LET K$ = INKEY$
150 IF K$=""THEN GO TO 310
155 PRINT AT 5,8;B
160 IF CODE K$-28<>B THEN GO TO 280
170 PRINT AT 5,15;"CORRECT"
180 LET S=S+1
190 IF S<>6 THEN GO TO 220
200 PRINT AT 6,0;"YOUbGOTb";CO;"bGOLDbCOINSb+b";100
    "bFROGS"
210 STOP
220 LET X=5*S
230 FOR J=0 TO 2
```

```
240 PRINT AT J,X-5;"bbb"
250 NEXT J
260 GO SUB 5ØØ
270 GO TO 29Ø
280 PRINT AT 5,15;"NO.bANSWERbISb";B
300 PAUSE 3ØØ
310 LET CO=CO-1
315 IF CO=Ø THEN GO TO 34Ø
320 PRINT AT 3,29;"b";CO
325 IF CO<1Ø THEN PRINT "b"
326 IF K$=""THEN GO TO 14Ø
330 NEXT I
340 PRINT AT 3,3;"..........................."
350 PRINT AT 6,Ø;"HARDbLUCKb-bYOUbWILLbGETbWET"
360 STOP
500 PRINT AT Ø,X+1;"O"
510 PRINT AT 1,X;"▀█▀"
520 PRINT AT 2,X;"▐b▌"
530 RETURN
```

List of Variables

S	=	number of stepping stone upon which man is standing
CO	=	number of gold coins left
X	=	column position of man
I	=	number of current question
A	=	)
B	=	) the question posed in the form A X B = C
C	=	)
J	=	loop counter

Comments

The child using the program need only press a single key to enter his answer – the use of INKEY$ removes the need for NEWLINE at the end of entries.

After each attempt the correct sum stays on the screen for six seconds and the coin transmutation also temporarily halts. This period can be cut short by pressing NEWLINE if required, since this terminates the PAUSE. If the man reaches the chest then the game ends with a message showing how many coins (and frogs!) he obtains.

To make the program easy to use a number of PRINT statements giving instructions on play should be included at the beginning of the listing.

Exercise 3(a): The theme of the program — crossing a river to a Treasure Chest — could be used in many different tasks other than a maths quiz, or at varying levels of difficulty. Modify the program to apply to a subject area of your choice.

SPELLING BIG WORDS (16K)

Description

This program is a spelling test which works by means of a word being displayed with a missing letter and the child has to enter the letter within a time limit. The word is displayed as four times normal size using direct access to the monitor character table. Words are entered by the teacher in a separate part of the program with the letter to be omitted being entered as an inverse character. N.B. Words up to 8 letters.

Method

Teacher :

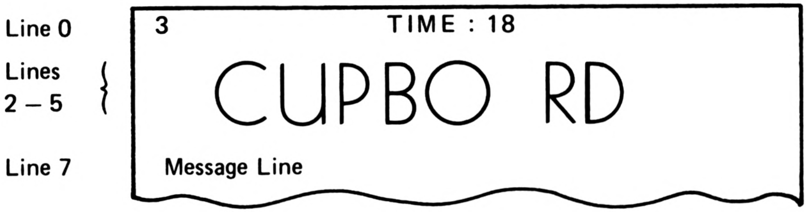
- (i) For I from 1 to 1Ø
  Enter word number I
- (ii) Stop

Child:

- (i) Set score S to Ø
- (ii) For I from 1 to 1Ø
  - a) Display word I, large with letter omitted
  - b) Set counter to 2Ø
  - c) Display counter
  - d) If no key pressed
    - 1) Decrement counter
    - 2) Display counter
    - 3) If counter = Ø, display message & go to (g)
    - 4) Go to (d) above
  - e) If key pressed is correct
    - 1) Add 1 to S
    - 2) Display message
    - 3) Go to (g)

f) Display message and show correct letter
g) Pause
(iii) Display score S and stop

Screen Format



Program Listing (16K)

```
10  DIM W$(10,8)
20  PRINT TAB 7;"TEACHERSbSECTION"
30  PRINT "ENTERbWORDSbWITHbLETTERbTObBE"
40  PRINT "OMITTEDbINbINVERSEbFORM"
50  FOR I=1 TO 10
60  PRINT AT I+5,5;"WORD";I;TAB 12;"=";
70  INPUT W$(I)
80  PRINT W$(I)
90  NEXT I
100 STOP
210 LET S=0
220 FOR I=1 TO 10
225 CLS
226 FAST
227 PRINT AT 0,0;I
230 FOR K=1 TO 8
240 LET C=CODE W$(I,K)
250 IF C<128 THEN GO TO 280
260 LET M$=CHR$(C-128)
270 LET C=0
280 FOR L=0 TO 7
290 LET P=PEEK(7680+C*8+L)
300 LET V=128
```

```
310 FOR J =0 TO 7
320 IF P<V THEN GO TO 350
330 PLOT 8*(K-1)+J,39-L
340 LET P=P-V
350 LET V=V/2
360 NEXT J
370 NEXT L
380 NEXT K
390 SLOW
400 LET X=20
410 PRINT AT 0,13;"TIME:20"
420 LET K$=INKEY$
430 IF K$<>""THEN GO TO 490
440 LET X=X-1
450 PRINT AT 0,18;X;"b"
460 IF X<>0 THEN GO TO 420
470 PRINT AT 7,0;"TOObSLOW"
480 GO TO 540
490 IF K$<>M$ THEN GO TO 530
500 LET S=S+1
510 PRINT AT 7,0;"CORRECT."
520 GO TO 540
530 PRINT AT 7,0;"WRONG."
540 PRINT AT 7,10;"IT bWASb";W$(I)
550 PAUSE 300
560 POKE 16437,255
570 NEXT I
580 CLS
590 PRINT AT 7,5;"YOUbSCOREDb";S;"bOUTbOFb10"
```

List of Variables

W$	=	array holding ten eight-character words
I	=	number of word currently considered
S	=	score out of ten
K	=	letter of current word
C	=	code of letter in word
M$	=	missing letter in word
L	=	loop counter
P	=	value of location in character table
V	=	a power of two used to access bits in P

J = loop counter
X = time counter
K$ = key pressed by player

Comments

The large display of the word being tested helps to give the program considerable visual impact and the single key entry of answers is also useful. The quiz proper is started by GOTO 21Ø.

There is a delay while the ZX81 sets up the large word on the screen and this takes place with a blank screen (uses FAST mode) in order that the player only has a given time limit to choose his answer: he does not see the word gradually appearing on the display. The routine is explained on page 102. Display of large characters is applicable to many areas of language teaching.

Exercise 3(b): Write statements to ensure that only valid entries are permitted in the teacher's section.

SPOTS BEFORE THE EYES (1K)

Description

Here is a ZX81 version of a program which first appeared in *'The ZX80 Companion'* under the title of PATTERN RECOGNITION. The idea behind the program is based upon Glenn Doman's book *'Teach Your Baby Maths'*, in which it is suggested that children can be taught to recognise quite large numbers of dots (up to one hundred) with sufficient practice. The program makes use of the PAUSE instruction to display a collection of spots for a very short time before the user enters the number. There are ten goes and a score is given at the end. The program runs on a 1K ZX81.

Method

(i) Set scores S, T and U to zero
(ii) For go number G from 1 to 1Ø
  a) Choose R at random between 2Ø and 1ØØ
  b) Display R spots for 2 seconds

c) Enter user's attempt N
d) If N=R (1) display "CORRECT"
(2) add 1 to S
(3) go to (h) below
e) If $|N-R| \leqslant 5$,(1) add 1 to T
(2) go to (g) below
f) If $|N-R| \leqslant 1Ø$, add 1 to U
g) Display "INCORRECT" and value of R
h) Wait for five seconds

(iii) Display scores.

Program Listing (1K)

```
 1Ø   LET S=Ø
 2Ø   LET T=Ø
 3Ø   LET U=Ø
 4Ø   FOR G=1 TO 1Ø
 5Ø   LET R=2Ø+ INT(RND*81)
 6Ø   FAST
 7Ø   FOR I=1 TO R
 8Ø   PRINT "◰";
 85   IF RND<Ø.5 THEN PRINT "b";
 9Ø   NEXT I
1ØØ   PAUSE 1ØØ
1Ø5   POKE 16437,255
11Ø   CLS
12Ø   SLOW
13Ø   PRINT "HOWbMANY?b";
14Ø   INPUT N
145   PRINT N
15Ø   IF N<>R THEN GO TO 19Ø
16Ø   PRINT "YES" (inverse)
17Ø   LET S=S+1
18Ø   GO TO 25Ø
19Ø   IF ABS(N-R)>5 THEN GO TO 22Ø
2ØØ   LET T=T+1
21Ø   GO TO 24Ø
22Ø   IF ABS(N-R)>1Ø THEN GO TO 24Ø
23Ø   LET U=U+1
24Ø   PRINT "NO,b";R
25Ø   PAUSE 25Ø
255   POKE 16437,255
```

```
26Ø    CLS
27Ø    NEXT G
28Ø    PRINT "SCORE:bb";S;"bCORRECT,b";T;"bWITHINb5,
       bb";U;"bWITHINb1Ø"
```

List of Variables

S	=	score correct
T	=	score within 5
U	=	score within 1Ø
G	=	go number, 1--1Ø
R	=	number of spots
I	=	loop counter
N	=	user's attempt

Comments

Note that the addition of line 85 ensures that the spots appear in a random pattern – without it the user can use the length of the pattern to estimate the number. The score given at the end in three parts (correct, within 5 and within 1Ø) gives the user a clear idea of his performance.

The program would run perfectly well in FAST Mode, but the listing above reverts to SLOW mode for the input section as a matter of the author's preference! The display of the spots **must** be in FAST mode so that the two second display of the spots is effective.

> **Exercise 3(c):** Why is the ABS function included in lines 19Ø and 22Ø and what would happen if it was omitted?

GRAB THE GRUNGER (16K)

Description

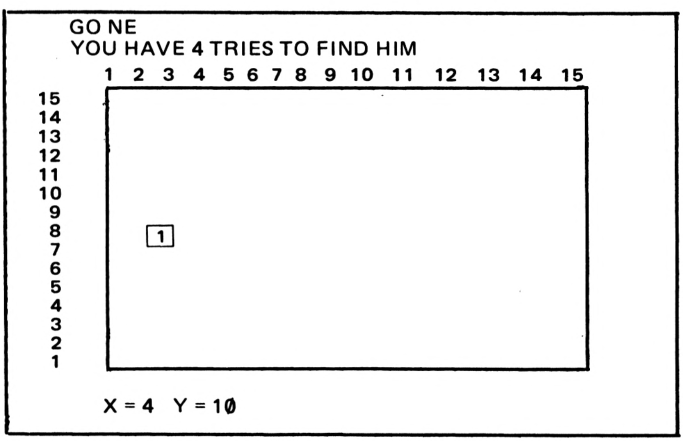
No discussion of primary level educational programs is complete without looking at grid games, of the HUNT THE HURKLE variety. In such programs a grid is displayed on the screen and an object (an imaginary creature such as a Hurkle, or in this case a GRUNGER) is chosen to be at a random position in the grid. The player then has a number of guesses to find the object and in some variations the object may move or even try and find the player. The educational value lies in that the player has to specify grid positions by means of standard X and Y axis co-ordinates, and the program gives the player hints by means of specifying the direction in which to move to find the object.

The following program uses a 15 x 15 grid in which the dreaded Grunger is hiding. The player has five tries to find it and after each attempt the program tells the player in which direction to proceed, e.g. SOUTH-EAST. The game ends and repeats when the player finds the Grunger or when he runs out of guesses.

Method

(i) Display instructions and grid with labelled axis
(ii) Choose random position of Grunger as (A,B)
(iii) For go number I from 1 to 5
  (a) Enter user's version of position, (X,Y)
  (b) If (X,Y) = (A,B)
    (1) Display CORRECT
    (2) Go to (v) below
  (c) Display direction to move
(iv) Show Grunger's position
(v) Wait
(vi) Go to (i)

Screen Format



Program Listing

```
1Ø  PRINT TAB 5;"AbGRUNGERbISbHIDINGbIN", TAB 8;"Ab
    15bXb15 GRID","YOUbHAVEb5bTRIESbTObFINDbHIM"
2Ø  FOR I = 15 TO 1 STEP -1
3Ø  PRINT TAB 6;I;TAB 9;"
4Ø  NEXT I
5Ø  PRINT TAB 9;"123456789111111bX"
6Ø  PRINT TAB 18; "012345"
7Ø  PRINT AT 1Ø, 1; "Y"
8Ø  LET A = INT (RND*15) + 1
9Ø  LET B = INT (RND*15) + 1
1ØØ FOR I = 1 TO 5
11Ø PRINT AT 2,9;6-I
12Ø PAUSE 5ØØ
13Ø POKE 16437,255
14Ø PRINT AT 21,Ø;"bbbbbbbbbbbbbbbbbbb"
15Ø PRINT AT 21,Ø;I;")bX=";
16Ø INPUT X
17Ø IF X<1 OR X>15 THEN GO TO 16Ø
18Ø PRINT X;",Y=";
19Ø INPUT Y
2ØØ IF Y<1 OR Y>15 THEN GO TO 19Ø
21Ø PRINT Y
22Ø PRINT AT 15-Y+3, X+8;I
23Ø IF X=A AND Y=B THEN GO TO 5ØØ
24Ø PRINT AT Ø,5;"bbbbbbbbbbbbbbbbbbbbb",TAB 8;
    "bbbbbbbbbbbbb"
25Ø PRINT AT Ø,13;"GOb"; CHR$ (51*(Y<B) + 56*(Y>B));
    CHR$(42*(X<A) + 60* (X>A))
26Ø NEXT I
27Ø PRINT AT Ø,7;"SORRYb-bITbWASb",A;",";B
28Ø PRINT AT 15-B + 3,A + 8; "G"
29Ø GO TO 51Ø
5ØØ PRINT AT Ø,12; "CORRECT"
51Ø PRINT AT 2,Ø;"bbbbbbbbbbbbbbbbbbbbbbbbbbbbbb"
52Ø PAUSE 9ØØØ
53Ø POKE 16437,255
54Ø CLS
55Ø RUN
```

List of Variables

A = x position of Grunger
B = y position of Grunger
X = user's guess, x position
Y = user's guess, y position
I = loop counter and go number

Comments

Notice how the number of tries left is shown at the top of the screen, as part of the original playing instructions. To modify the program for 1K, omit the display of the grid, i.e. lines 2Ø–7Ø, 22Ø,28Ø and modify lines 14Ø and 15Ø to use screen line 4 instead of line 21.

Exercise 3 (d): What size grid can be designed for a 1K ZX81 t include a grid display?

COPYCAT (1K)

Description

Here is a straight forward memory test. A sequence of letters is displayed, one letter at a time, on the ZX81 screen and the player has to repeat the sequence afterwards. The sequence starts at three letters but goes up to twenty! Letters are displayed eight times normal size for clarity, using a PRINT AT version of the routine used in BIG WORDS on page 74. At the end the player is given a mark showing the maximum number of letters he has copied correctly. The program wor on a 1K ZX81.

Method

(i) Generate 2Ø random letters in A$
(ii) For no. of letters I from 3 to 2Ø
  (a) For letter number J from 1 to I
    1) Display letter J of A$, large
    2) Wait for two seconds
    3) Clear screen
  (b) Enter user's version of sequence, X$

(c) If correct
  1) Display "RIGHT SO FAR . . ."
  2) Wait for five seconds
(d) If incorrect display message, correct sequence and score (I–1)

(iii) Display "CONGRATULATIONS"

Program Listing (1K)

```
 10 DIM A$(20)
 20 FOR I=1 TO 20
 30 LET A$(I)=CHR$(INT(RND*27)+38)
 40 NEXT I
 50 FOR I=3 TO 20
 55 CLS
 60 FOR J=1 TO I
 70 LET C=CODE(A$(J))
 80 FOR H=0 TO 7
 90 LET P=PEEK(7680+C*8+H)
100 LET V=128
110 FOR G=0 TO 7
120 IF P<V THEN GO TO 150
130 PRINT AT H,G;"█"
140 LET P=P-V
150 LET V=V/2
160 NEXT G
170 NEXT H
180 PAUSE 100
186 POKE 16437,255
190 CLS
200 NEXT J
210 PRINT "SEQUENCE=";
220 INPUT X$
225 PRINT X$
230 IF X$<>A$(1 TO I) THEN GO TO 300
240 PRINT "RIGHT SO FAR ..." (inverse)
250 PAUSE 250
260 POKE 16437,255
270 NEXT I
280 PRINT "CONGRATULATIONS" (inverse)
290 STOP
300 PRINT "NOb-b";A$(1 TO I);".bYOUbGOTb";I-1
```

List of Variables

A$	=	array holding 2Ø character sequence
I	=	loop counter and count showing length of current sequence
J	=	counter showing character in current sequence
H	=	counter indicating appropriate number of byte in character table
P	=	value of byte in character table
V	=	a power of 2
G	=	counter showing current bit being tested
X$	=	player's version of sequence
C	=	character code of current character

Comments

By changing the randomising instruction at line 3Ø the program can easily be modified to handle sequences of numbers, or even graphics symbols.

Exercise 3(e): Change line 3Ø to produce sequences of digits Ø to 9 rather than letters.

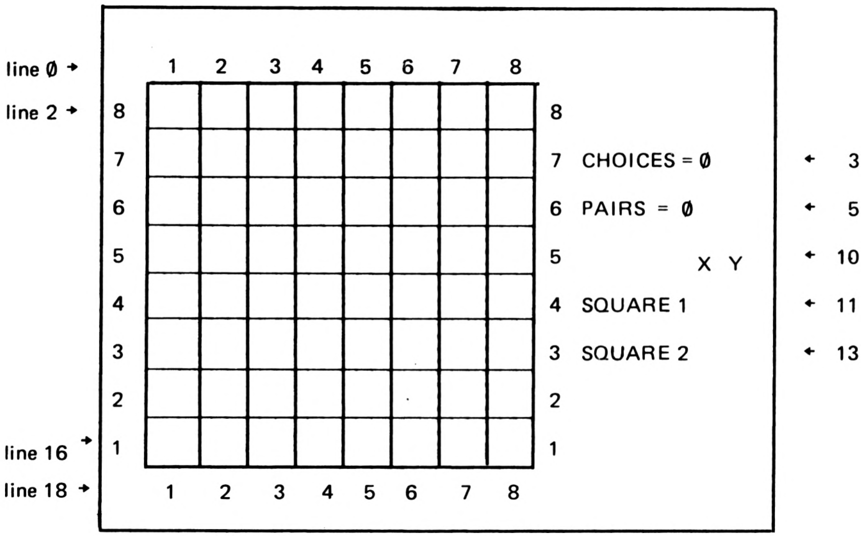
PICKING PAIRS (16K)

Description

A useful exercise of memory whether by a child or an adult is the game of Concentration, in which a number of cards are shuffled and l out singly face down. The player then has to choose a pair of cards, look at them, and if they are a pair of the same type (e.g. Aces, Three they are left face up. Otherwise they are turned face down again and another pair chosen. This continues until all the cards are face up. Clearly the player has to try and remember the positions of cards tha has seen.

This program works on the same basis, using a grid sized 8 x 8 filled v eight sets of the letters A to H. The player chooses a pair by specifyi a pair of column (X) and row (Y) positions. If an identical pair is fou the letters stay on the screen, whereas if the letters chosen are differe they disappear after ten seconds. Running totals of choices and pairs are displayed on the screen. Entry of positions is done by single key depressions, NEWLINE not being needed (i.e. INKEY$ is used) and there is built-in error checking.

Screen Format



Program Listing

```
 5 DIM B(4)
1Ø DIM A$(8,8)
15 PRINT "SETTINGbUP";
2Ø FOR I=1 TO 8
3Ø LET A$(I)="ABCDEFGH"
4Ø NEXT I
45 FOR I=1 TO 1ØØ
47 IF I=1Ø*INT(I/1Ø) THEN PRINT ".";
5Ø FOR J=1 TO 4
55 LET B(J)=INT(RND*8)+1
6Ø NEXT J
65 LET X$=A$(B(1),B(2))
7Ø LET A$(B(1),B(2))=A$(B(3),B(4))
```

```
 75 LET A$(B(3),B(4))=X$
 8Ø NEXT I
 85 CLS
 9Ø PRINT
 95 FOR I=1 TO 8
1ØØ PRINT "b█████████████████"
1Ø5 PRINT "b█b█b█b█b█b█b█b█b█"
1Ø7 NEXT I
11Ø PRINT "b█████████████████"
115 PRINT AT 18,2;"1b2b3b4b5b6b7b8"
12Ø PRINT AT Ø,2;"1b2b3b4b5b6b7b8"
13Ø FOR L=2 TO 16 STEP 2
14Ø PRINT AT L,Ø;9-L/2;TAB 18; 9-L/2
15Ø NEXT L
16Ø LET P=Ø
17Ø LET O=Ø
18Ø PRINT AT 3,21;"CHOICES = Ø"
19Ø PRINT AT 5,22;"PAIRS=Ø"
2ØØ PRINT AT 1Ø,29;"XbY"
21Ø PRINT AT 11,2Ø;"SQUARE 1"
22Ø PRINT AT 13,2Ø;"SQUARE 2"
23Ø PRINT AT 13,29;"bbb"
232 PRINT AT 11,29;"bbb"
235 LET L=11
24Ø LET C=29
25Ø GO SUB 6ØØ
26Ø LET X1=K
27Ø LET C=31
28Ø GO SUB 6ØØ
29Ø LET Y1=K
3ØØ IF CODE (A$(X1,Y1))>128 THEN GO TO 23Ø
31Ø LET L=13
32Ø LET C=29
33Ø GO SUB 6ØØ
34Ø LET X2=K
35Ø LET C=31
36Ø GO SUB 6ØØ
37Ø LET Y2=K
38Ø IF CODE (A$(X2,Y2))>128 THEN GO TO 23Ø
39Ø PRINT AT 18-2*Y1,2*X1;A$(X1,Y1)
4ØØ PRINT AT 18-2*Y2,2*X2;A$(X2,Y2)
41Ø LET O=O+1
```

```
420 PRINT AT 3,29;O
430 IF A$(X1,Y1)=A$(X2,Y2) THEN GO TO 480
440- PAUSE 500
450 PRINT AT 18-2*Y1,2*X1;"b"
460 PRINT AT 18-2*Y2,2*X2;"b"
470 GO TO 230
480 LET P=P+1
490 PRINT AT 5,28;P
500 LET A$(X1,Y1)=CHR$(CODE(A$(X1,Y1))+128)
510 LET A$(X2,Y2)=CHR$(CODE(A$(X2,Y2))+128)
520 IF P<>32 THEN GO TO 230
530 PRINT AT 15,22;"WELLbDONE" (inverse)
540 PRINT AT 2,29;"▄▄"
550 PRINT AT 4,29;"▀▀"
560 STOP
600 PRINT AT L,C;"?" (inverse)
610 LET K$ = INKEY$
620 IF K$<="8" AND K$>="1" THEN GO TO 650
630 PRINT AT L,C;"?"
640 GO TO 600
650 LET K=VAL K$
660 PRINT AT L,C;K
670 RETURN
```

List of Variables

B = array of four random numbers used to shuffle A$
A$ = 8 x 8 array of characters
I = loop counter
J = loop counter
L = line number counter
C = column number counter
P = no. of identical pairs chosen
O = no. of choices made
X1,Y1 = coordinates of first member of pair
X2,Y2 = coordinates of second member of pair
K$ = value of key pressed
K = entry of x or y position

Comments

The program demonstrates several interesting features. The subroutines at line 6ØØ shows how an input prompt may be highlighted by a 'blinking' question mark and then accepted without NEWLINE, i.e. using INKEY$.

The eight sets of eight letters are initially put into A$ in sequence and then shuffled using one hundred random exchanges.

Exercise 3(f): How can you crash the program while it is waiting for an x or y input? Modify the program to overcome this.

PRIMES (1K)

Background

Having considered a number of primary level programs we now look at higher things. As mentioned above, the best educational programs are written by subject specialists, so the author's intention with the rest of the chapter is to illustrate some techniques for readers to apply to their own areas. The topic chosen is mathematics, and the next two programs are *demonstrations* to illustrate mathematical concepts or techniques. An excellent reference for mathematical computer programs is "A Collection of Programming Problems and Techniques" by Maurer and Williams, published by Prentice-Hall, Inc.

There is a slight problem when doing complex maths on the ZX81, as illustrated by

```
PRINT 2Ø  –  Ø.ØØØØØØØØ1
```

which does not give 19.999999999 or even 2Ø but 52. Yes, there is a bug in the ZX81's floating point arithmetic which leaps out when handlir numbers of considerably different magnitudes. Beware!

Description

The following program accepts any number greater than one and calculates and prints the number's factors if any, or indicates that it is a prime number.

Method

(i) Enter number M
(ii) If M less than two, stop
(iii) Set N to M
(iv) Set divisor D to 2
(v) If N is divisible by D
    a) Display D
    b) Divide N by D
    c) Go to (v)
(vi) Add 1 to D
(vii) If D less than M go to (v)
(viii) If N>1 display "PRIME"
(ix) Wait
(x) Go to (i)

Program Listing (1K)

```
  5 CLS
 10 PRINT "ENTERbNUMBERb";
 20 INPUT M
 30 IF M<2 THEN STOP
 40 LET N=M
 50 PRINT N
 60 LET D=2
 70 IF N<>D*INT(N/D) THEN GO TO 100
 80 PRINT D;"b";
 85 LET N=N/D
 90 GO TO 70
100 LET D=D+1
110 IF D<M THEN GO TO 70
120 IF N>1 THEN PRINT "PRIME"
130 IF N=1 THEN PRINT "AREbTHEbFACTORS"
140 PAUSE 9999
150 RUN
```

List of Variables

M = number as input
N = number divided by factor(s)
D = divisor

Exercise 3(g): Modify the above program so that when calculating whether m is prime, the highest possible factor used is the square root of m.

ITERATION (1K)

Description

The ZX81 is an excellent tool for demonstrating simple iterative techniques for the solution of equations. The Newton Raphson method is used to solve an equation of the form

$$f(x) = 0$$

by taking an initial approximation x_o to the solution $x=\alpha$ and successive improving it by generating a sequence:

$$x_o \; x_1 \; x_2 \; x_3 \; \ldots$$

which converges to the solution.

The iterative formula is

$$x_{n+1} = x_n - \frac{f(x_n)}{f^1(x_n)}$$

Where $f'(x)$ is the differential of $f(x)$.
We determine whether we have reached the solution by considering successive approximations : if two approximations x_j and x_{j+1} fulfill:

$|x_j - x_{j+1}| < \epsilon$ where ϵ is a small constant then we are close enough. ϵ is chosen by the user according to the accuracy required.

The program accepts an equation of up to the fifth order and given an initial approximation, calculates a solution.

Method

(i) Enter order of equation N
(ii) For counter I from N+1 to 1

(a) Enter coefficient A(I) (for a_{i+1} in $a_{i+1}x^i$)

(iii) Enter approximation XA

(iv) Calculate array B, the differential coefficients.

(v) Calculate and print next approximation XB using coefficients in A and B

(vi) If XA–XB <Ø.ØØØØ1 then

(a) Display solution

(b) Stop

(vii) Set XA to XB

(viii) Go to (v)

Program Listing

```
10Ø  DIM A(6)
3Ø   PRINT "ORDER=";
4Ø   INPUT N
5Ø   IF N>5 OR N<1 THEN GO TO 4Ø
6Ø   LET N=INT(N)
65   PRINT N
7Ø   FOR I=N+1 TO 1 STEP –1
75   SCROLL
8Ø   PRINT AT 1,Ø;"COEFFTbOFbX**";I–1;"=";
9Ø   INPUT A(I)
1ØØ  PRINT A(I)
11Ø  NEXT I
12Ø  PRINT "APPROX=";
13Ø  INPUT XA
14Ø  PRINT XA
15Ø  LET C=2
18Ø  LET F=A(1)
19Ø  FOR I=2 TO N+1
2ØØ  LET F=F+A(I)*XA**(I–1)
21Ø  NEXT I
22Ø  LET DF=A(2)
23Ø  FOR I=2 TO N
24Ø  LET DF=DF+A(I+1)*I*XA**(I–1)
25Ø  NEXT I
26Ø  LET XB=XA–F/DF
265  SCROLL
27Ø  PRINT AT 1,Ø;"APPROXb";C;"=";XB
28Ø  LET C=C+1
29Ø  IF ABS(XA–XB)< Ø.ØØØØ1 THEN GO TO 32Ø
```

```
300 LET XA=XB
310 GO TO 180
320 PRINT "SOLUTION=";XB
```

List of Variables

A = array holding coefficients of powers of x
e.g. $f(x) = a_6x^5 + a_5x^4 + a_4x^3 + a_3x^2 + a_2x + a_1$
N = order of f(x) i.e. highest power of x
I = loop counter
XA =) successive approximations
XB)
F = f(x) at XA
DF = f'(x) at XA
C = number of approximations

Comments

The program almost fills a 1K ZX81 and there is very little room left for a screen display. Therefore if using a 16K machine, extend the number of approximations displayed, i.e. change the PRINT AT instructions to use a line around 15 or so.

Exercise 3(h): In what circumstances would line 260 terminate with an error? Modify the program to overcome this.

THE QUIZ (16 K)

Description

At the beginning of the chapter we reviewed types of educational programs. This program is a general purpose quiz in which the teacher can set up a bank of questions and answers on any topic (and in any language!) and the ZX81 then poses the questions to a student in the form of an interactive quiz. There are two notable points about this program: firstly a standard question format is entered by the teacher

e.g. WHAT IS THE FORMULA FOR

or TRANSLATE INTO SWAHILI
or WHAT IS THE CAPITAL OF

and then pairs of question and answer keywords make up the remainder; secondly, a student's answer is marked correct providing it **contains** the answer keyword

e.g. if the answer keyword is OPTIC

then all the following responses are correct

OPTICAL ISOMERISM
OPTIC
OPTICALLY
CHANGES OPTICALLY
OPTICALLISH

Naturally enough, this is a 16K program.

Method

TEACHER:
- (i) Enter number of questions Q
- (ii) Enter maximum length L1 of question word and L2 of answer word.
- (iii) Enter form of question F$
- (iv) For counter I from 1 to Q
  - (a) Enter question word number I into Q$(I)
  - (b) Enter answer word number I into A$(I)
  - (c) Store length of A$(I) in A(I)
- (v) Stop

STUDENT:
- (i) Set score S to Ø
- (ii) For counter I from 1 to Q
  - (a) Display question F$ and question word Q$(I)
  - (b) Enter student's response X$
  - (c) If X$ contains answer word A$(I)
    - 1) Display CORRECT
    - 2) Add 1 to 5
    - 3) Go to (e) below
  - (d) Display WRONG & answer word A$(I)
  - (e) Repeat

(iii) Display score S

Program Listing

```
10 PRINT "NO.bOFbQUESTIONS"
20 INPUT Q
30 IF Q<5 OR Q >50 THEN GO TO 20
40 PRINT Q
50 PRINT "MAX.bLENGTHbOFbQbWORD=";
60 INPUT L1
70 IF L1<1 OR L1>30 THEN GO TO 60
80 PRINT L1
90 PRINT TAB 15;"AbWORD=";
100 INPUT L2
110 IF L2<1 OR L2>30 THEN GO TO 100
120 PRINT L2
122 DIM Q$(Q,L1)
124 DIM A$(Q,L2)
126 DIM A(Q)
130 PRINT "QUESTIONbFORMAT="
140 INPUT F$
150 IF F$="" THEN GO TO 140
160 PRINT F$
165 PAUSE 250
170 CLS
180 FOR I=1 TO Q
182 SCROLL
185 SCROLL
190 PRINT AT 18,0;"Q";I;"=";
200 INPUT Q$(I)
210 PRINT Q$(I)
220 PRINT "A";I;"=";
230 INPUT A$(I)
231 FOR J=L2 TO 1 STEP -1
232 IF A$(I,J)<>"b" THEN GO TO 234
233 NEXT J
234 LET A(I)=J
240 PRINT A$(I)
270 NEXT I
280 PRINT "ENDbOFbINPUT"
290 STOP
300 LET S=0
```

```
31Ø FOR I=1 TO Q
32Ø CLS
33Ø PRINT I;")b";F$
34Ø PRINT Q$(I)
35Ø INPUT X$
355 PRINT X$
36Ø IF LEN X$<A(I) THEN GO TO 4ØØ
37Ø FOR J=1 TO LEN X$–A(I)+1
38Ø IF X$(J TO J+A(I)–1)=A$(I,1 TO A(I)) THEN GO TO 42Ø
39Ø NEXT J
4ØØ PRINT "NOb–bANSWERbISb";A$(I)
41Ø GO TO 44Ø
42Ø PRINT "CORRECT"  (inverse)
43Ø LET S=S+1
44Ø PAUSE 5ØØ
45Ø NEXT I
46Ø CLS
47Ø PRINT "SCORE:b";S;"bOUTbOFb";Q
```

List of Variables

Q	=	number of questions
L1	=	maximum length of question words
L2	=	maximum length of answer words
Q$	=	questions
A$	=	answers
A	=	array holding actual lengths of answer words
F$	=	question format
X$	=	student's answer
S	=	score

Comments

The teacher sets up a bank of questions and answers having started the program by RUN. The student uses the program by GO TO 3ØØ, and each of the questions appear in turn. Providing the student's response to a question **contains** the answer keyword, it is marked correct. A score appears at the end.

Exercise 3(i): What is the purpose of line 36Ø and what would happen if it was omitted?

CHAPTER FOUR – THE MONITOR

4.1 EXAMINING AND USING THE MONITOR

Introduction

This chapter aims to introduce readers to the way in which the 8K ROM Monitor is organised, how it may be examined and how it may be used. Much of the chapter is taken up by a listing of the contents of the ROM in terms of tables of data and assembly language instructions. In order to understand in any detail the workings of the Monitor a knowledge of Z80 low level language is required, but readers without this knowledge will find that parts of the chapter illustrating data tables in the monitor or describing start addresses of Monitor routines will be useful. A good book for learning about the Z80 is Rodnay Zacs' "Programming the Z80". Readers should see Chapters 24 - 27 of the Sinclair Manual for further background information.

Hexadecimal

As described in Chapter 24 of the Sinclair Manual, binary and hexadecim numbering is generally used when discussing the contents of ZX81 memory locations. Consider a location containing the decimal number 28. In binary this is

Ø	Ø	Ø	1	1	1	Ø	Ø
2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0

since $2^4 + 2^3 + 2^2 = 16 + 8 + 4 = 28$

In hexadecimal we have

1C hex = 28 decimal

But taking each hex digit as four binary digits

$$\overbrace{Ø\;Ø\;Ø\;1}^{1}\qquad\overbrace{1\;1\;Ø\;Ø}^{C}$$

thus showing how hexadecimal is a useful "shorthand" for binary.

In this chapter we will be using both decimal and hexadecimal numbers to represent memory addresses and contents. Therefore a useful start is a program to convert decimal numbers to hexadecimal.

The program uses an algorithm based upon a manual method of conversion. Consider for example 7654 decimal. If we successively divide this by 16 and take the remainders we have

```
16 \ 7654
  16 \ 478   remainder 6    = 6 hex
    16 \ 29   remainder 14  = E hex
      16 \ 1   remainder 13 = D hex
           0    remainder 1 = 1 hex

        Thus 7654 decimal   =      1   D   E   6   hex
```

Here is the program:

```
  5 DIM H$(4)
 10 PRINT "NUMBER=";
 20 INPUT C
 25 IF C=0 THEN STOP
 30 GO SUB 500
 40 PRINT C;"bbHEX=";
 50 IF C>255 THEN PRINT H$(1);H$(2);
 60 PRINT H$(3);H$(4)
 70 PRINT
 80 GO TO 10
500 LET D1=C
510 FOR I=4 TO 1 STEP -1
520 LET D2=INT(D1/16)
530 LET H$(I)=CHR$(D1-16*D2+28)
540 LET D1=D2
550 NEXT I
560 RETURN
```

The subroutine at line 500 does the conversion to hexadecimal while the first part of the program enters a number and then prints a hexadecimal number of an appropriate size.

Conversion from hexadecimal to decimal is simpler.

```
 10 PRINT "HEXbNUMBER=";
 20 INPUT H$
 30 IF H$=""THEN STOP
 40 LET D=CODE H$(1)-28
 50 FOR I=2 TO LEN H$
 60 LET D=16*D + CODE H$(I)-28
 70 NEXT I
 80 PRINT H$;"b,DECIMAL=";D
 90 PRINT
100 GO TO 10
```

Monitor Routines and Entry Points

The disassembled listing of the 8K monitor given in Section 4.2 gives readers a chance to work out for themselves just how the ZX81 works. To make the task a little bit simpler the following points will be helpful. Addresses given below are in hexadecimal.

(i) The program starts are location ØØØØ as in any Z80 system.

(ii) RST ØØØ8 is the 'error report handling' entry point.
It is entered by using the instruction 'CF – RST ØØØ8' followed by a data byte for the required error., e.g. see
Ø2F4 RST ØØØ8
Ø2F5 'ØE'
which gives error 'F'.

(iii) RST ØØ1Ø is the character printing routine. The normal way to print a character to the next position on the screen is to load the A register with the appropriate character code (including NEWLINE) and then call this routine by using the instruction 'D7 – RST ØØ1Ø.'

(iv) RST ØØ18 and RST ØØ2Ø are routines for collecting the next character in a BASIC line.

(v) RST ØØ28 is the entry point for the 'floating-point calculator', which starts at location 199C. (See note xxxvii).

(vi) RST ØØ3Ø is a routine that will make 'BC' spaces in the variable area.

(vii) RST ØØ38 is the interrupt routine that handles the lines of the T.V. display.

(viii) The routine at ØØ66 is the NMI routine that leads to a T.V. display being formed following a NM interrupt in 'slow' mode.

(ix) The main 'key table' is at ØØ7E to ØØCB. There is a code for each key in 'lower'case and in 'shift'.

(x) The key-codes for the 'function mode' are in the table from ØØCC to ØØF2.

(xi) The key-codes for the 'graphics mode' are in the table from ØØF3 to Ø11Ø.

(xii) The command table is at Ø111 to Ø1FB. Each keyword is listed with its last letter inverted.

(xiii) The 'update routine' at Ø1FC to Ø2Ø6 is used by the LOAD and SAVE command routines.

(xiv) The routines from Ø2Ø7 to Ø2BA are used to produce the T.V. display.

(xv) The keyboard scanning routine at Ø2BB to Ø2E6 is a very useful routine. Each key of the keyboard gives a unique key-value in the HL register pair. No key pressed gives the value FFFF.

(xvi) The SAVE command routine is at Ø2F6 to Ø33F.

(xvii) The LOAD command routine is at Ø34Ø to Ø3A7.

(xviii) The routine at Ø3CB to Ø3E4 is the RAM integrity check routine that is carried out upon initialisation and following a NEW command.

(xix) The main initialisation routine starts at Ø3E5, and is followed by the operating system routines for handling the 'cursor' and forming LISTings.

(xx) The main command routine for the running of a BASIC program is from Ø63E to Ø6DF.

(xxi) The keyboard decode routine at Ø7B4 to Ø7DB is also very useful as it converts the key-values (in BC now) to the values 1–78 and forms the appropriate address, in HL, for a given key in the main key table. (see note ix).

(xxii) The routine at Ø7F1 to Ø868 is the character printing routine used by RST ØØ1Ø. (see note iii.)

(xxiii) The routine at Ø8F5 to Ø94A is concerned with expanding the display file, in the case of a 'collapsed' display file. The routine in effect sets the system variable 'DF–CC' to a legitimate address.

(xxiv) The CLS command routine is at ØA2A to ØA5F.

(xxv) The PRINT command routine is at ØACF to ØBAE.

(xxvi) The PLOT/UNPLOT command routine is at ØBAF to ØCØD. The difference between the commands being dependant on the current value of T–ADDR.

(xxvii) The SCROLL command routine is at ØCØE to ØC28.

(xxciii) The main syntax tables are at ØC29 to ØCB9. The first part being a pointer table and the second part the actual syntax table that gives the required syntax for each command and the address of the 'command routine'.

(xxix) The BASIC interpreter starts at ØCBA.

(xxx) The FAST command routine is at ØF2Ø to ØF27 and can be simply called using 'CALL ØF2Ø' to enter FAST mode, or ensure the presence in FAST mode.

(xxxi) The SLOW command routine is at ØF28 to ØF2E and can likewise be called by 'CALL ØF28'.

(xxxii) The 'Expression Evaluator' starts at location ØF52.

(xxxiii) The LET command routine is at 131D to 14Ø4.

(xxxiv) The DIM command routine is at 14Ø5 to 1483.

(xxxv) The routines between 14CA and 1913 are concerned with handling 'floating-point' numbers. e.g. the routine 'Evaluate to integer' is at 1586. Print 'Last value' is at 15D7, etc.

(xxxvi) The function table for the 'floating-point calculator' is at 1914 to 199B.

(xxxvii) The 'floating-point calculator' is at 199C to 1AA8.

(xxxviii) The various function routines are at 1AA9 to 1DFF. e.g. CHR$ is at 1B8E to 1BA2, COS is at 1D3D to 1D47, etc.

(xxxix) The 'character generator' is at 1EØØ to 1FFF. This part of the 8K ROM holds the 8*8 formats of the 64 characters that can appear on the T.V. display.

Program Aids

A number of BASIC programs can be written to assist in the examination of the Monitor, and particularly the data tables.

HEX DISPLAY

This program displays the contents of Monitor addresses in hexadecimal starting at a specified address:

```
 5 DIM H$(4)
1Ø PRINT "START=";
2Ø INPUT S
25 PRINT S
3Ø FOR A=S TO 8191 STEP 8
35 SCROLL
```

```
 40 LET C=A
 50 GO SUB 500
 60 PRINT AT 5,0;H$;"bbb";
 70 FOR B=A TO A+7
 80 LET C=PEEK(B)
 90 GO SUB 500
100 PRINT H$(3 TO 4);"b";
110 NEXT B
120 PRINT
130 NEXT A
140 STOP
500 LET D1=C
510 FOR 1=4 TO 1 STEP -1
520 LET D2=INT(D1/16)
530 LET H$(I)=CHR$(D1-16*D2+28)
540 LET D1=D2
550 NEXT I
560 RETURN
```

This works in 1K – for a 16K system change the PRINT AT statement at line 60 to give a larger screen display.

CHARACTER DISPLAY

This program displays the contents of Monitor addresses as characters – useful for some data tables.

```
10 PRINT "START=";
20 INPUT S
25 PRINT S
30 FOR A=S TO 8191
40 SCROLL
50 PRINT AT 15,0;A;"bbb";CHR$ (PEEK A)
60 NEXT A
```

The program is very handy for displaying the Key Table (locations 126 to 272) and the following Command Table (locations 273 to 507). As described in the previous section the Key Table holds codes for the keyboard keys in 'lower' case, in shifted form, in function mode and finally in graphics mode. RUN the program with start address equalling 126 and the appropriate keyboard values will be shown. In the Command

Table we find each keyword with the last letter held in inverse form. To show this, run the program with a start address of 273, or simply let it run on after the Key Table.

CHARACTER GENERATOR DISPLAY

The last data table in the Monitor is the character generator held in locations 768Ø to 8191. This holds the formats for each of the 64 characters used on the ZX81 by means of eight bytes per character, each byte consisting of Ø's and 1's representing unshaded and shaded portions respectively.

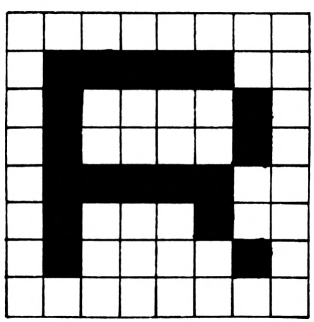
For example the letter R has character code 55. The portion of the character table holding the format for R is locations 812Ø to 8127.

i.e 768Ø +(55 x 8) to 768Ø + (55 x 8) + 7

The binary patterns in these locations are shown below

Location	Contents
812Ø	Ø Ø Ø Ø Ø Ø Ø Ø
8121	Ø 1 1 1 1 1 Ø Ø
8122	Ø 1 Ø Ø Ø Ø 1 Ø
8123	Ø 1 Ø Ø Ø Ø 1 Ø
8124	Ø 1 1 1 1 1 Ø Ø
8125	Ø 1 Ø Ø Ø 1 Ø Ø
8126	Ø 1 Ø Ø Ø Ø 1 Ø
8127	Ø Ø Ø Ø Ø Ø Ø Ø

This represents



A routine to use the character table to display large characters has already been utilised in Chapter Three, and is shown again below:

To display characters at four times size we use the PLOT statement.

```
 10 PRINT "CHARACTER=";
 20 INPUT A$
 25 LET C=CODE A$
 30 PRINT A$;"bCODE=";C;"ATbLOCb";7680+C*8
 40 FOR H=0 TO 7
 50 LET P=PEEK(7680+C*8+H)
 60 LET V=128
 70 FOR G=0 TO 7
 80 IF P<V THEN GO TO 110
 90 PLOT G,40-H
100 LET P=P-V
110 LET V=V/2
120 NEXT G
130 NEXT H
```

For eight times size we use the PRINT AT statement and this can help illustrate the unshaded portions. We display the shaded portions using "■" and the unshaded by "▩". Modify the above by

```
 80 IF P<V THEN GO TO 108
 90 PRINT AT H+3, G;"■"
104 GO TO 110
108 PRINT AT H+3,G;"▩"
```

and the character will appear black on a grey background.

4.2 MONITOR LISTING

The next eleven pages contain a disassembled listing of the ZX81 8K ROM Monitor between addresses 0000 and 0CB9, that is, up to the end of the syntax table. The rest of the Monitor has not been included since much of it consists of the BASIC interpreter which is not particularly interesting or usable.

A description of Monitor routines and tables appears in Section 4.1 and the listing should be studied in conjunction with this.

```
0000  D3 FD         OUT (FD),A
0002  01 FF 7F      LD BC,7FFF
0005  C3 CB 03      JP 03CB
0008  2A 16 40      LD HL,(4016)
000B  22 18 40      LD (4018),HL
000E  18 46         JR 0056
0010  A7            AND A
0011  C2 F1 07      JP NZ,07F1
0014  C3 F5 07      JP 07F5
0017  FF            RST 38
0018  2A 16 40      LD HL,(4016)
001B  7E            LD A,(HL)
001C  A7            AND A
001D  C0            RET NZ
001E  00            NOP
001F  00            NOP
0020  CD 49 00      CALL 0049
0023  18 F7         JR 001C
0025  FF            RST 38
0026  FF            RST 38
0027  FF            RST 38
0028  C3 9C 19      JP 199C
002B  F1            POP AF
002C  D9            EXX
002D  E3            EX (SP),HL
002E  D9            EXX
002F  C9            RET
0030  C5            PUSH BC
0031  2A 14 40      LD HL,(4014)
0034  E5            PUSH HL
0035  C3 84 14      JP 1484
0038  0D            DEC C
0039  C2 45 00      JP NZ,0045
003C  E1            POP HL
003D  05            DEC B
003E  C8            RET Z
003F  CB D9         SET 3,C
0041  ED 4F         LD R,A
0043  FB            EI
0044  E9            JP (HL)
0045  D1            POP DE
0046  C8            RET Z
0047  18 F8         JR 0041
0049  2A 16 40      LD HL,(4016)
004C  23            INC HL
004D  22 16 40      LD (4016),HL
0050  7E            LD A,(HL)
0051  FE 7F         CP 7F
0053  C0            RET NZ
0054  18 F6         JR 004C
0056  E1            POP HL
0057  6E            LD L,(HL)
0058  FD 75 00      LD (IY),L
005B  ED 7B 02 40   LD SP,(4002)
005F  CD 07 02      CALL 0207
0062  C3 B8 14      JP 14B8
0065  FF            RST 38
0066  08            EX AF,AF'
0067  3C            INC A
0068  FA 6D 00      JP M,006D
006B  28 02         JR Z,006F
006D  08            EX AF,AF'
006E  C9            RET
006F  08            EX AF,AF'
0070  F5            PUSH AF
0071  C5            PUSH BC
0072  D5            PUSH DE
0073  E5            PUSH HL
0074  2A 0C 40      LD HL,(400C)
0077  CB FC         SET 7,H
0079  76            HALT
007A  D3 FD         OUT (FD),A
007C  DD E9         JP (IX)
007E                                                  3F 3D
      28 3B 26 38  29 2B 2C 36  3C 2A 37 39  1D 1E 1F 20
      21 1C 25 24  23 22 35 34  2E 3A 3E 76  31 30 2F 20
      00 1B 32 33  27 0E 19 0F  18 E3 E1 E4  E5 E2 C0 D9
      E0 DB DD 75  DA DE DF 72  77 74 73 70  71 0B 11 10
      0D DC 79 14  15 16 D8 0C  1A 12 13 17  CD CE C1 78
      CA CB CC D1  D2 C7 C8 C9  CF 40 78 78  78 78 78 78
      78 78 78 78  C2 D3 C4 D6  D5 78 D4 C6  C5 D0 78 78
      42 D7 41 08  0A 09 8A 89  81 82 07 84  06 01 02 87
      04 05 77 78  85 03 83 8B  91 90 8D 86  78 92 95 96
      88 8F 0B 8B  26 B9 39 26  A7 8F 28 34  29 AA 3B 26
      B1 31 2A B3  38 2E B3 28  34 B8 39 26  B3 26 38 B3
      26 28 B8 26  39 B3 31 B3  2A 3D B5 2E  33 B9 38 36
      B7 38 2C B3  26 27 B8 35  2A 2A B0 3A  38 B7 38 39
      37 8D 28 2D  37 8D 33 34  B9 17 97 34  B7 26 33 A9
      13 94 12 94  13 92 39 2D  2A B3 39 B4  38 39 2A B5
      31 35 37 2E  33 B9 31 31  2E 38 B9 38  39 34 B5 38
      31 34 BC 2B  26 38 B9 33  2A BC 38 28  37 34 31 B1
      28 34 33 B9  29 2E B2 37  2A B2 2B 34  B7 2C 34 39
      B4 2C 34 38  3A A7 2E 33  35 3A B9 31  34 26 A9 31
      2E 38 B9 31  2A B9 35 26  3A 38 AA 33  2A 3D B9 35
      34 30 AA 35  37 2E 33 B9  35 31 34 B9  37 3A B3 38
      26 3B AA 37  26 33 A9 2E  AB 28 31 B8  3A 33 35 31
      34 B7 28 31  2A 26 B7 37  2A 39 3A 37  B3 28 34 35
      BE 37 33 A9  2E 33 30 2A  3E 8D 35 AE
```

```
01FC  23            INC HL
01FD  EB            EX DE,HL
01FE  2A 14 40      LD HL,(4014)
0201  37            SCF
0202  ED 52         SBC HL,DE
0204  EB            EX DE,HL
0205  D0            RET NC
0206  E1            POP HL
0207  21 3B 40      LD HL,403B
020A  7E            LD A,(HL)
020B  17            RLA
020C  AE            XOR (HL)
020D  17            RLA
020E  D0            RET NC
020F  3E 7F         LD A,7F
0211  08            EX AF,AF'
0212  06 11         LD B,11
0214  D3 FE         OUT (FE),A
0216  10 FE         DJNZ 0216
0218  D3 FD         OUT (FD),A
021A  08            EX AF,AF'
021B  17            RLA
021C  30 08         JR NC,0226
021E  CB FE         SET 7,(HL)
0220  F5            PUSH AF
0221  C5            PUSH BC
0222  D5            PUSH DE
0223  E5            PUSH HL
0224  18 03         JR 0229
0226  CB B6         RES 6,(HL)
0228  C9            RET
0229  2A 34 40      LD HL,(4034)
022C  2B            DEC HL
022D  3E 7F         LD A,7F
022F  A4            AND H
0230  B5            OR L
0231  7C            LD A,H
0232  20 03         JR NZ,0237
0234  17            RLA
0235  18 02         JR 0239
0237  46            LD B,(HL)
0238  37            SCF
0239  67            LD H,A
023A  22 34 40      LD (4034),HL
023D  D0            RET NC
023E  CD BB 02      CALL 02BB
0241  ED 4B 25 40   LD BC,(4025)
0245  22 25 40      LD (4025),HL
0248  78            LD A,B
0249  C6 02         ADD A,02
024B  ED 42         SBC HL,BC
024D  3A 27 40      LD A,(4027)
0250  B4            OR H
0251  B5            OR L
0252  58            LD E,B
0253  06 0B         LD B,0B
0255  21 3B 40      LD HL,403B
0258  CB 86         RES 0,(HL)
025A  20 08         JR NZ,0264
025C  CB 7E         BIT 7,(HL)
025E  CB C6         SET 0,(HL)
0260  C8            RET Z
0261  05            DEC B
0262  00            NOP
0263  37            SCF
0264  21 27 40      LD HL,4027
0267  3F            CCF
0268  CB 10         RL B
026A  10 FE         DJNZ 026A
026C  46            LD B,(HL)
026D  7B            LD A,E
026E  FE FE         CP FE
0270  9F            SBC A,A
0271  06 1F         LD B,1F
0273  B6            OR (HL)
0274  A0            AND B
0275  1F            RRA
0276  77            LD (HL),A
0277  D3 FF         OUT (FF),A
0279  2A 0C 40      LD HL,(400C)
027C  CB FC         SET 7,H
027E  CD 92 02      CALL 0292
0281  ED 5F         LD A,R
0283  01 01 19      LD BC,1901
0286  3E F5         LD A,F5
0288  CD B5 02      CALL 02B5
028B  2B            DEC HL
028C  CD 92 02      CALL 0292
028F  C3 29 02      JP 0229
0292  DD E1         POP IX
0294  FD 4E 28      LD C,(IY+28)
0297  FD CB 3B 7E   BIT 7,(IY+3B)
029B  28 0C         JR Z,02A9
029D  79            LD A,C
029E  ED 44         NEG
02A0  3C            INC A
02A1  08            EX AF,AF'
02A2  D3 FE         OUT (FE),A
02A4  E1            POP HL
02A5  D1            POP DE
02A6  C1            POP BC
02A7  F1            POP AF
02A8  C9            RET
02A9  3E FC         LD A,FC
02AB  06 01         LD B,01
02AD  CD B5 02      CALL 02B5
02B0  2B            DEC HL
02B1  E3            EX (SP),HL
02B2  E3            EX (SP),HL
02B3  DD E9         JP (IX)
02B5  ED 4F         LD R,A
02B7  3E DD         LD A,DD
02B9  FB            EI
02BA  E9            JP (HL)
02BB  21 FF FF      LD HL,FFFF
02BE  01 FE FE      LD BC,FEFE
02C1  ED 78         IN A,(C)
02C3  F6 01         OR 01
02C5  F6 E0         OR E0
02C7  57            LD D,A
02C8  2F            CPL
02C9  FE 01         CP 01
02CB  9F            SBC A,A
02CC  B0            OR B
02CD  A5            AND L
02CE  6F            LD L,A
02CF  7C            LD A,H
02D0  A2            AND D
02D1  67            LD H,A
02D2  CB 00         RLC B
02D4  ED 78         IN A,(C)
02D6  38 ED         JR C,02C5
02D8  1F            RRA
02D9  CB 14         RL H
02DB  17            RLA
02DC  17            RLA
02DD  17            RLA
02DE  9F            SBC A,A
02DF  E6 18         AND 18
02E1  C6 1F         ADD A,1F
02E3  32 28 40      LD (4028),A
```

```
02E6  C9           RET
02E7  FD CB 3B 7E  BIT 7,(IY+3B)
02EB  C8           RET Z
02EC  76           HALT
02ED  D3 FD        OUT (FD),A
02EF  FD CB 3B BE  RES 7,(IY+3B)
02F3  C9           RET
02F4  CF           RST 8
02F5  0E CD        LD C,CD
02F7  A8           XOR B
02F8  03           INC BC
02F9  38 F9        JR C,02F4
02FB  EB           EX DE,HL
02FC  11 CB 12     LD DE,12CB
02FF  CD 43 0F     CALL 0F43
0302  30 2E        JR NC,0332
0304  10 FE        DJNZ 0304
0306  1B           DEC DE
0307  7A           LD A,D
0308  B3           OR E
0309  20 F4        JR NZ,02FF
030B  CD 1E 03     CALL 031E
030E  CB 7E        BIT 7,(HL)
0310  23           INC HL
0311  28 F8        JR Z,030B
0313  21 09 40     LD HL,4009
0316  CD 1E 03     CALL 031E
0319  CD FC 01     CALL 01FC
031C  18 F8        JR 0316
031E  5E           LD E,(HL)
031F  37           SCF
0320  CB 13        RL E
0322  C8           RET Z
0323  9F           SBC A,A
0324  E6 05        AND 05
0326  C6 04        ADD A,04
0328  4F           LD C,A
0329  D3 FF        OUT (FF),A
032B  06 23        LD B,23
032D  10 FE        DJNZ 032D
032F  CD 43 0F     CALL 0F43
0332  30 72        JR NC,03A6
0334  06 1E        LD B,1E
0336  10 FE        DJNZ 0336
0338  0D           DEC C
0339  20 EE        JR NZ,0329
033B  A7           AND A
033C  10 FD        DJNZ 033B
033E  18 E0        JR 0320
0340  CD A8 03     CALL 03A8
0343  CB 12        RL D
0345  CB 0A        RRC D
0347  CD 4C 03     CALL 034C
034A  18 FB        JR 0347
034C  0E 01        LD C,01
034E  06 00        LD B,00
0350  3E 7F        LD A,7F
0352  DB FE        IN A,(FE)
0354  D3 FF        OUT (FF),A
0356  1F           RRA
0357  30 49        JR NC,03A2
0359  17           RLA
035A  17           RLA
035B  38 28        JR C,0385
035D  10 F1        DJNZ 0350
035F  F1           POP AF
0360  BA           CP D
0361  D2 E5 03     JP NC,03E5
0364  62           LD H,D
0365  6B           LD L,E
0366  CD 4C 03     CALL 034C
0369  CB 7A        BIT 7,D
036B  79           LD A,C
036C  20 03        JR NZ,0371
036E  BE           CP (HL)
036F  20 D6        JR NZ,0347
0371  23           INC HL
0372  17           RLA
0373  30 F1        JR NC,0366
0375  FD 34 15     INC (IY+15)
0378  21 09 40     LD HL,4009
037B  50           LD D,B
037C  CD 4C 03     CALL 034C
037F  71           LD (HL),C
0380  CD FC 01     CALL 01FC
0383  18 F6        JR 037B
0385  D5           PUSH DE
0386  1E 94        LD E,94
0388  06 1A        LD B,1A
038A  1D           DEC E
038B  DB FE        IN A,(FE)
038D  17           RLA
038E  CB 7B        BIT 7,E
0390  7B           LD A,E
0391  38 F5        JR C,0388
0393  10 F5        DJNZ 038A
0395  D1           POP DE
0396  20 04        JR NZ,039C
0398  FE 56        CP 56
039A  30 B2        JR NC,034E
039C  3F           CCF
039D  CB 11        RL C
039F  30 AD        JR NC,034E
03A1  C9           RET
03A2  7A           LD A,D
03A3  A7           AND A
03A4  28 BB        JR Z,0361
03A6  CF           RST 8
03A7  0C           INC C
03A8  CD 52 0F     CALL 0F52
03AB  3A 01 40     LD A,(4001)
03AE  87           ADD A,A
03AF  FA 9A 0D     JP M,0D9A
03B2  E1           POP HL
03B3  D0           RET NC
03B4  E5           PUSH HL
03B5  CD E7 02     CALL 02E7
03B8  CD F4 13     CALL 13F4
03BB  62           LD H,D
03BC  6B           LD L,E
03BD  0D           DEC C
03BE  F8           RET M
03BF  09           ADD HL,BC
03C0  CB FE        SET 7,(HL)
03C2  C9           RET
03C3  CD E7 02     CALL 02E7
03C6  FD 4B 04 40  LD BC,(4004)
03CA  0B           DEC BC
03CB  60           LD H,B
03CC  69           LD L,C
03CD  3E 3F        LD A,3F
03CF  36 02        LD (HL),02
03D1  2B           DEC HL
03D2  BC           CP H
03D3  20 FA        JR NZ,03CF
03D5  A7           AND A
03D6  ED 42        SBC HL,BC
03D8  09           ADD HL,BC
03D9  23           INC HL
03DA  30 06        JR NC,03E2
03DC  35           DEC (HL)
03DD  28 03        JR Z,03E2
03DF  35           DEC (HL)
03E0  28 F3        JR Z,03D5
```

```
03E2  22 04 40      LD (4004),HL
03E5  2A 04 40      LD HL,(4004)
03E8  2B            DEC HL
03E9  36 3E         LD (HL),3E
03EB  2B            DEC HL
03EC  F9            LD SP,HL
03ED  2B            DEC HL
03EE  2B            DEC HL
03EF  22 02 40      LD (4002),HL
03F2  3E 1E         LD A,1E
03F4  ED 47         LD I,A
03F6  ED 56         IM1
03F8  FD 21 00 40   LD IY,4000
03FC  FD 36 3B 40   LD (IY+3B),40
0400  21 7D 40      LD HL,407D
0403  22 0C 40      LD (400C),HL
0406  06 19         LD B,19
0408  36 76         LD (HL),76
040A  23            INC HL
040B  10 FB         DJNZ 0408
040D  22 10 40      LD (4010),HL
0410  CD 96 14      CALL 1496
0413  CD A9 14      CALL 14A9
0416  CD 07 02      CALL 0207
0419  CD 2A 0A      CALL 0A2A
041C  2A 0A 40      LD HL,(400A)
041F  ED 5B 23 40   LD DE,(4023)
0423  A7            AND A
0424  ED 52         SBC HL,DE
0426  EB            EX DE,HL
0427  30 04         JR NC,042D
0429  19            ADD HL,DE
042A  22 23 40      LD (4023),HL
042D  CD D8 09      CALL 09D8
0430  28 01         JR Z,0433
0432  EB            EX DE,HL
0433  CD 3E 07      CALL 073E
0436  FD 35 1E      DEC (IY+1E)
0439  20 37         JR NZ,0472
043B  2A 0A 40      LD HL,(400A)
043E  CD D8 09      CALL 09D8
0441  2A 16 40      LD HL,(4016)
0444  37            SCF
0445  ED 52         SBC HL,DE
0447  21 23 40      LD HL,4023
044A  30 0B         JR NC,0457
044C  EB            EX DE,HL
044D  7E            LD A,(HL)
044E  23            INC HL
044F  ED A0         LDI
0451  12            LD (DE),A
0452  18 C5         JR 0419
0454  21 0A 40      LD HL,400A
0457  5E            LD E,(HL)
0458  23            INC HL
0459  56            LD D,(HL)
045A  E5            PUSH HL
045B  EB            EX DE,HL
045C  23            INC HL
045D  CD D8 09      CALL 09D8
0460  CD BB 05      CALL 05BB
0463  E1            POP HL
0464  FD CB 2D 6E   BIT 5,(IY+2D)
0468  20 08         JR NZ,0472
046A  72            LD (HL),D
046B  2B            DEC HL
046C  73            LD (HL),E
046D  18 AA         JR 0419
046F  CD A9 14      CALL 14A9
0472  2A 14 40      LD HL,(4014)
0475  7E            LD A,(HL)
0476  FE 7E         CP 7E
0478  20 08         JR NZ,0482
047A  01 06 00      LD BC,0006
047D  CD 60 0A      CALL 0A60
0480  18 F3         JR 0475
0482  FE 76         CP 76
0484  23            INC HL
0485  20 EE         JR NZ,0475
0487  CD 37 05      CALL 0537
048A  CD 1F 0A      CALL 0A1F
048D  2A 14 40      LD HL,(4014)
0490  FD 36 00 FF   LD (IY),FF
0494  CD 66 07      CALL 0766
0497  FD CB 00 7E   BIT 7,(IY)
049B  20 24         JR NZ,04C1
049D  3A 22 40      LD A,(4022)
04A0  FE 18         CP 18
04A2  30 1D         JR NC,04C1
04A4  3C            INC A
04A5  32 22 40      LD (4022),A
04A8  47            LD B,A
04A9  0E 01         LD C,01
04AB  CD 18 09      CALL 0918
04AE  54            LD D,H
04AF  5D            LD E,L
04B0  7E            LD A,(HL)
04B1  2B            DEC HL
04B2  BE            CP (HL)
04B3  20 FC         JR NZ,04B1
04B5  23            INC HL
04B6  EB            EX DE,HL
04B7  3A 05 40      LD A,(4005)
04BA  FE 4D         CP 4D
04BC  DC 5D 0A      CALL C,0A5D
04BF  18 C9         JR 048A
04C1  21 00 00      LD HL,0000
04C4  22 18 40      LD (4018),HL
04C7  21 3B 40      LD HL,403B
04CA  CB 7E         BIT 7,(HL)
04CC  CC 29 02      CALL Z,0229
04CF  CB 46         BIT 0,(HL)
04D1  28 FC         JR Z,04CF
04D3  ED 4B 25 40   LD BC,(4025)
04D7  CD 4B 0F      CALL 0F4B
04DA  CD BD 07      CALL 07BD
04DD  30 93         JR NC,0472
04DF  3A 06 40      LD A,(4006)
04E2  3D            DEC A
04E3  FA 08 05      JP M,0508
04E6  20 0F         JR NZ,04F7
04E8  32 06 40      LD (4006),A
04EB  1D            DEC E
04EC  7B            LD A,E
04ED  D6 27         SUB 27
04EF  38 01         JR C,04F2
04F1  5F            LD E,A
04F2  21 CC 00      LD HL,00CC
04F5  18 0E         JR 0505
04F7  7E            LD A,(HL)
04F8  FE 76         CP 76
04FA  28 2F         JR Z,052B
04FC  FE 40         CP 40
04FE  CB FF         SET 7,A
0500  38 19         JR C,051B
0502  21 C7 00      LD HL,00C7
0505  19            ADD HL,DE
0506  18 0D         JR 0515
0508  7E            LD A,(HL)
0509  FD CB 01 56   BIT 2,(IY+01)
050D  20 07         JR NZ,0516
050F  C6 C0         ADD A,C0
0511  FE E6         CP E6
0513  30 01         JR NC,0516
```

```
0515  7E            LD A,(HL)
0516  FE F0         CP F0
0518  EA 2D 05      JP PE,052D
051B  5F            LD E,A
051C  CD 37 05      CALL 0537
051F  7B            LD A,E
0520  CD 26 05      CALL 0526
0523  C3 72 04      JP 0472
0526  CD 9B 09      CALL 099B
0529  12            LD (DE),A
052A  C9            RET
052B  3E 78         LD A,78
052D  5F            LD E,A
052E  21 B2 04      LD HL,04B2
0531  19            ADD HL,DE
0532  19            ADD HL,DE
0533  4E            LD C,(HL)
0534  23            INC HL
0535  46            LD B,(HL)
0536  C5            PUSH BC
0537  2A 14 40      LD HL,(4014)
053A  FD CB 2D 6E   BIT 5,(IY+2D)
053E  20 16         JR NZ,0556
0540  FD CB 01 96   RES 2,(IY+01)
0544  7E            LD A,(HL)
0545  FE 7F         CP 7F
0547  C8            RET Z
0548  23            INC HL
0549  CD B4 07      CALL 07B4
054C  28 F6         JR Z,0544
054E  FE 26         CP 26
0550  38 F2         JR C,0544
0552  FE DE         CP DE
0554  28 EA         JR Z,0540
0556  FD CB 01 D6   SET 2,(IY+01)
055A  18 E8         JR 0544
055C  01 01 00      LD BC,0001
055F  C3 60 0A      JP 0A60
0562  9F            SBC A,A
0563  05            DEC B
0564  54            LD D,H
0565  04            INC B
0566  76            HALT
0567  05            DEC B
0568  7F            LD A,A
0569  05            DEC B
056A  AF            XOR A
056B  05            DEC B
056C  C4 05 0C      CALL NZ,0C05
056F  06 8B         LD B,8B
0571  05            DEC B
0572  AF            XOR A
0573  05            DEC B
0574  AF            XOR A
0575  05            DEC B
0576  CD 93 05      CALL 0593
0579  7E            LD A,(HL)
057A  36 7F         LD (HL),7F
057C  23            INC HL
057D  18 09         JR 0588
057F  23            INC HL
0580  7E            LD A,(HL)
0581  FE 76         CP 76
0583  28 18         JR Z,059D
0585  36 7F         LD (HL),7F
0587  2B            DEC HL
0588  77            LD (HL),A
0589  18 98         JR 0523
058B  CD 93 05      CALL 0593
058E  CD 5C 05      CALL 055C
0591  18 F6         JR 0589
0593  2B            DEC HL
0594  ED 5B 14 40   LD DE,(4014)
0598  1A            LD A,(DE)
0599  FE 7F         CP 7F
059B  C0            RET NZ
059C  D1            POP DE
059D  18 EA         JR 0589
059F  2A 0A 40      LD HL,(400A)
05A2  CD D8 09      CALL 09D8
05A5  EB            EX DE,HL
05A6  CD BB 05      CALL 05BB
05A9  21 0B 40      LD HL,400B
05AC  C3 64 04      JP 0464
05AF  7B            LD A,E
05B0  E6 07         AND 07
05B2  32 06 40      LD (4006),A
05B5  18 E6         JR 059D
05B7  EB            EX DE,HL
05B8  11 C2 04      LD DE,04C2
05BB  7E            LD A,(HL)
05BC  E6 C0         AND C0
05BE  20 F7         JR NZ,05B7
05C0  56            LD D,(HL)
05C1  23            INC HL
05C2  5E            LD E,(HL)
05C3  C9            RET
05C4  CD 1F 0A      CALL 0A1F
05C7  21 6F 04      LD HL,046F
05CA  E5            PUSH HL
05CB  FD CB 2D 6E   BIT 5,(IY+2D)
05CF  C0            RET NZ
05D0  2A 14 40      LD HL,(4014)
05D3  22 0E 40      LD (400E),HL
05D6  21 21 18      LD HL,1821
05D9  22 39 40      LD (4039),HL
05DC  2A 0A 40      LD HL,(400A)
05DF  CD D8 09      CALL 09D8
05E2  CD BB 05      CALL 05BB
05E5  7A            LD A,D
05E6  B3            OR E
05E7  C8            RET Z
05E8  2B            DEC HL
05E9  CD A5 0A      CALL 0AA5
05EC  23            INC HL
05ED  4E            LD C,(HL)
05EE  23            INC HL
05EF  46            LD B,(HL)
05F0  23            INC HL
05F1  ED 5B 0E 40   LD DE,(400E)
05F5  3E 7F         LD A,7F
05F7  12            LD (DE),A
05F8  13            INC DE
05F9  E5            PUSH HL
05FA  21 1D 00      LD HL,001D
05FD  19            ADD HL,DE
05FE  09            ADD HL,BC
05FF  ED 72         SBC HL,SP
0601  E1            POP HL
0602  D0            RET NC
0603  ED B0         LDIR
0605  EB            EX DE,HL
0606  D1            POP DE
0607  CD A2 14      CALL 14A2
060A  18 91         JR 059D
060C  CD 1F 0A      CALL 0A1F
060F  21 72 04      LD HL,0472
0612  FD CB 2D 6E   BIT 5,(IY+2D)
0616  20 11         JR NZ,0629
0618  2A 14 40      LD HL,(4014)
061B  7E            LD A,(HL)
061C  FE FF         CP FF
061E  28 06         JR Z,0626
0620  CD E2 08      CALL 08E2
```

```
0623  CD 2A 0A      CALL 0A2A
0626  21 19 04      LD HL,0419
0629  E5            PUSH HL
062A  CD BA 0C      CALL 0CBA
062D  E1            POP HL
062E  CD 37 05      CALL 0537
0631  CD 5C 05      CALL 055C
0634  CD 73 0A      CALL 0A73
0637  20 15         JR NZ,064E
0639  78            LD A,B
063A  B1            OR C
063B  C2 E0 06      JP NZ,06E0
063E  0B            DEC BC
063F  0B            DEC BC
0640  ED 43 07 40   LD (4007),BC
0644  FD 36 22 02   LD (IY+22),02
0648  ED 5B 0C 40   LD DE,(400C)
064C  18 13         JR 0661
064E  FE 76         CP 76
0650  28 12         JR Z,0664
0652  ED 4B 30 40   LD BC,(4030)
0656  CD 18 09      CALL 0918
0659  ED 5B 29 40   LD DE,(4029)
065D  FD 36 22 02   LD (IY+22),02
0661  DF            RST 18
0662  FE 76         CP 76
0664  CA 13 04      JP Z,0413
0667  FD 36 01 80   LD (IY+01),80
066B  EB            EX DE,HL
066C  22 29 40      LD (4029),HL
066F  EB            EX DE,HL
0670  CD 4D 00      CALL 004D
0673  CD C1 0C      CALL 0CC1
0676  FD CB 01 8E   RES 1,(IY+01)
067A  3E C0         LD A,C0
067C  FD 77 19      LD (IY+19),A
067F  CD 9F 14      CALL 149F
0682  FD CB 2D AE   RES 5,(IY+2D)
0686  FD CB 00 7E   BIT 7,(IY)
068A  28 22         JR Z,06AE
068C  2A 29 40      LD HL,(4029)
068F  A6            AND (HL)
0690  20 1C         JR NZ,06AE
0692  56            LD D,(HL)
0693  23            INC HL
0694  5E            LD E,(HL)
0695  ED 53 07 40   LD (4007),DE
0699  23            INC HL
069A  5E            LD E,(HL)
069B  23            INC HL
069C  56            LD D,(HL)
069D  23            INC HL
069E  EB            EX DE,HL
069F  19            ADD HL,DE
06A0  CD 43 0F      CALL 0F43
06A3  38 C7         JR C,066C
06A5  21 00 40      LD HL,4000
06A8  CB 7E         BIT 7,(HL)
06AA  28 02         JR Z,06AE
06AC  36 0C         LD (HL),0C
06AE  FD CB 38 7E   BIT 7,(IY+38)
06B2  CC 71 08      CALL Z,0871
06B5  01 21 01      LD BC,0121
06B8  CD 18 09      CALL 0918
06BB  3A 00 40      LD A,(4000)
06BE  ED 4B 07 40   LD BC,(4007)
06C2  3C            INC A
06C3  28 0C         JR Z,06D1
06C5  FE 09         CP 09
06C7  20 01         JR NZ,06CA
06C9  03            INC BC
06CA  ED 43 2B 40   LD (402B),BC
06CE  20 01         JR NZ,06D1
06D0  0B            DEC BC
06D1  CD EB 07      CALL 07EB
06D4  3E 18         LD A,18
06D6  D7            RST 10
06D7  CD 98 0A      CALL 0A98
06DA  CD A9 14      CALL 14A9
06DD  C3 C1 04      JP 04C1
06E0  ED 43 0A 40   LD (400A),BC
06E4  2A 16 40      LD HL,(4016)
06E7  EB            EX DE,HL
06E8  21 13 04      LD HL,0413
06EB  E5            PUSH HL
06EC  2A 1A 40      LD HL,(401A)
06EF  ED 52         SBC HL,DE
06F1  E5            PUSH HL
06F2  C5            PUSH BC
06F3  CD E7 02      CALL 02E7
06F6  CD 2A 0A      CALL 0A2A
06F9  E1            POP HL
06FA  CD D8 09      CALL 09D8
06FD  20 06         JR NZ,0705
06FF  CD F2 09      CALL 09F2
0702  CD 60 0A      CALL 0A60
0705  C1            POP BC
0706  79            LD A,C
0707  3D            DEC A
0708  B0            OR B
0709  C8            RET Z
070A  C5            PUSH BC
070B  03            INC BC
070C  03            INC BC
070D  03            INC BC
070E  03            INC BC
070F  2B            DEC HL
0710  CD 9E 09      CALL 099E
0713  CD 07 02      CALL 0207
0716  C1            POP BC
0717  C5            PUSH BC
0718  13            INC DE
0719  2A 1A 40      LD HL,(401A)
071C  2B            DEC HL
071D  ED B8         LDDR
071F  2A 0A 40      LD HL,(400A)
0722  EB            EX DE,HL
0723  C1            POP BC
0724  70            LD (HL),B
0725  2B            DEC HL
0726  71            LD (HL),C
0727  2B            DEC HL
0728  73            LD (HL),E
0729  2B            DEC HL
072A  72            LD (HL),D
072B  C9            RET
072C  FD CB 01 CE   SET 1,(IY+01)
0730  CD A7 0E      CALL 0EA7
0733  78            LD A,B
0734  E6 3F         AND 3F
0736  67            LD H,A
0737  69            LD L,C
0738  22 0A 40      LD (400A),HL
073B  CD D8 09      CALL 09D8
073E  1E 00         LD E,00
0740  CD 45 07      CALL 0745
0743  18 FB         JR 0740
0745  ED 4B 0A 40   LD BC,(400A)
0749  CD EA 09      CALL 09EA
074C  16 92         LD D,92
074E  28 05         JR Z,0755
0750  11 00 00      LD DE,0000
0753  CB 13         RL E
0755  FD 73 1E      LD (IY+1E),E
```

```
0758  7E           LD A,(HL)
0759  FE 40        CP 40
075B  C1           POP BC
075C  D0           RET NC
075D  C5           PUSH BC
075E  CD A5 0A     CALL 0AA5
0761  23           INC HL
0762  7A           LD A,D
0763  D7           RST 10
0764  23           INC HL
0765  23           INC HL
0766  22 16 40     LD (4016),HL
0769  FD CB 01 C6  SET 0,(IY+01)
076D  ED 4B 18 40  LD BC,(4018)
0771  2A 16 40     LD HL,(4016)
0774  A7           AND A
0775  ED 42        SBC HL,BC
0777  20 03        JR NZ,077C
0779  3E B8        LD A,B8
077B  D7           RST 10
077C  2A 16 40     LD HL,(4016)
077F  7E           LD A,(HL)
0780  23           INC HL
0781  CD B4 07     CALL 07B4
0784  22 16 40     LD (4016),HL
0787  28 E4        JR Z,076D
0789  FE 7F        CP 7F
078B  28 10        JR Z,079D
078D  FE 76        CP 76
078F  28 5D        JR Z,07EE
0791  CB 77        BIT 6,A
0793  28 05        JR Z,079A
0795  CD 4B 09     CALL 094B
0798  18 D3        JR 076D
079A  D7           RST 10
079B  18 D0        JR 076D
079D  3A 06 40     LD A,(4006)
07A0  06 AB        LD B,AB
07A2  A7           AND A
07A3  20 05        JR NZ,07AA
07A5  3A 01 40     LD A,(4001)
07A8  06 B0        LD B,B0
07AA  1F           RRA
07AB  1F           RRA
07AC  E6 01        AND 01
07AE  80           ADD A,B
07AF  CD F5 07     CALL 07F5
07B2  18 B9        JR 076D
07B4  FE 7E        CP 7E
07B6  C0           RET NZ
07B7  23           INC HL
07B8  23           INC HL
07B9  23           INC HL
07BA  23           INC HL
07BB  23           INC HL
07BC  C9           RET
07BD  16 00        LD D,00
07BF  CB 28        SRA B
07C1  9F           SBC A,A
07C2  F6 26        OR 26
07C4  2E 05        LD L,05
07C6  95           SUB L
07C7  85           ADD A,L
07C8  37           SCF
07C9  CB 19        RR C
07CB  38 FA        JR C,07C7
07CD  0C           INC C
07CE  C0           RET NZ
07CF  48           LD C,B
07D0  2D           DEC L
07D1  2E 01        LD L,01
07D3  20 F2        JR NZ,07C7
07D5  21 7D 00     LD HL,007D
07D8  5F           LD E,A
07D9  19           ADD HL,DE
07DA  37           SCF
07DB  C9           RET
07DC  7B           LD A,E
07DD  A7           AND A
07DE  F8           RET M
07DF  18 10        JR 07F1
07E1  AF           XOR A
07E2  09           ADD HL,BC
07E3  3C           INC A
07E4  38 FC        JR C,07E2
07E6  ED 42        SBC HL,BC
07E8  3D           DEC A
07E9  28 F1        JR Z,07DC
07EB  1E 1C        LD E,1C
07ED  83           ADD A,E
07EE  A7           AND A
07EF  28 04        JR Z,07F5
07F1  FD CB 01 86  RES 0,(IY+01)
07F5  D9           EXX
07F6  E5           PUSH HL
07F7  FD CB 01 4E  BIT 1,(IY+01)
07FB  20 05        JR NZ,0802
07FD  CD 08 08     CALL 0808
0800  18 03        JR 0805
0802  CD 51 08     CALL 0851
0805  E1           POP HL
0806  D9           EXX
0807  C9           RET
0808  57           LD D,A
0809  ED 4B 39 40  LD BC,(4039)
080D  79           LD A,C
080E  FE 21        CP 21
0810  28 1A        JR Z,082C
0812  3E 76        LD A,76
0814  BA           CP D
0815  28 30        JR Z,0847
0817  2A 0E 40     LD HL,(400E)
081A  BE           CP (HL)
081B  7A           LD A,D
081C  20 20        JR NZ,083E
081E  0D           DEC C
081F  20 19        JR NZ,083A
0821  23           INC HL
0822  22 0E 40     LD (400E),HL
0825  0E 21        LD C,21
0827  05           DEC B
0828  ED 43 39 40  LD (4039),BC
082C  78           LD A,B
082D  FD BE 22     CP (IY+22)
0830  28 03        JR Z,0835
0832  A7           AND A
0833  20 DD        JR NZ,0812
0835  2E 04        LD L,04
0837  C3 58 00     JP 0058
083A  CD 9B 09     CALL 099B
083D  EB           EX DE,HL
083E  77           LD (HL),A
083F  23           INC HL
0840  22 0E 40     LD (400E),HL
0843  FD 35 39     DEC (IY+39)
0846  C9           RET
0847  0E 21        LD C,21
0849  05           DEC B
084A  FD CB 01 C6  SET 0,(IY+01)
084E  C3 18 09     JP 0918
0851  FE 76        CP 76
0853  28 1C        JR Z,0871
0855  4F           LD C,A
0856  3A 38 40     LD A,(4038)
```

```
0859  E6 7F        AND 7F
085B  FE 5C        CP 5C
085D  6F           LD L,A
085E  26 40        LD H,40
0860  CC 71 08     CALL Z,0871
0863  71           LD (HL),C
0864  2C           INC L
0865  FD 75 38     LD (IY+38),L
0868  C9           RET
0869  16 16        LD D,16
086B  2A 0C 40     LD HL,(400C)
086E  23           INC HL
086F  18 05        JR 0876
0871  16 01        LD D,01
0873  21 3C 40     LD HL,403C
0876  CD E7 02     CALL 02E7
0879  C5           PUSH BC
087A  E5           PUSH HL
087B  AF           XOR A
087C  5F           LD E,A
087D  D3 FB        OUT (FB),A
087F  E1           POP HL
0880  CD 43 0F     CALL 0F43
0883  38 05        JR C,088A
0885  1F           RRA
0886  D3 FB        OUT (FB),A
0888  CF           RST 8
0889  0C           INC C
088A  DB FB        IN A,(FB)
088C  87           ADD A,A
088D  FA DE 08     JP M,08DE
0890  30 EE        JR NC,0880
0892  E5           PUSH HL
0893  D5           PUSH DE
0894  7A           LD A,D
0895  FE 02        CP 02
0897  9F           SBC A,A
0898  A3           AND E
0899  07           RLCA
089A  A3           AND E
089B  57           LD D,A
089C  4E           LD C,(HL)
089D  79           LD A,C
089E  23           INC HL
089F  FE 76        CP 76
08A1  28 24        JR Z,08C7
08A3  E5           PUSH HL
08A4  CB 27        SLA A
08A6  87           ADD A,A
08A7  87           ADD A,A
08A8  26 0F        LD H,0F
08AA  CB 14        RL H
08AC  83           ADD A,E
08AD  6F           LD L,A
08AE  CB 11        RL C
08B0  9F           SBC A,A
08B1  AE           XOR (HL)
08B2  4F           LD C,A
08B3  06 08        LD B,08
08B5  7A           LD A,D
08B6  CB 01        RLC C
08B8  1F           RRA
08B9  67           LD H,A
08BA  DB FB        IN A,(FB)
08BC  1F           RRA
08BD  30 FB        JR NC,08BA
08BF  7C           LD A,H
08C0  D3 FB        OUT (FB),A
08C2  10 F1        DJNZ 08B5
08C4  E1           POP HL
08C5  18 D5        JR 089C
08C7  DB FB        IN A,(FB)
08C9  1F           RRA
08CA  30 FB        JR NC,08C7
08CC  7A           LD A,D
08CD  0F           RRCA
08CE  D3 FB        OUT (FB),A
08D0  D1           POP DE
08D1  1C           INC E
08D2  CB 5B        BIT 3,E
08D4  28 A7        JR Z,087D
08D6  C1           POP BC
08D7  15           DEC D
08D8  20 A0        JR NZ,087A
08DA  3E 04        LD A,04
08DC  D3 FB        OUT (FB),A
08DE  CD 07 02     CALL 0207
08E1  C1           POP BC
08E2  21 5C 40     LD HL,405C
08E5  36 76        LD (HL),76
08E7  06 20        LD B,20
08E9  2B           DEC HL
08EA  36 00        LD (HL),00
08EC  10 FB        DJNZ 08E9
08EE  7D           LD A,L
08EF  CB FF        SET 7,A
08F1  32 38 40     LD (4038),A
08F4  C9           RET
08F5  3E 17        LD A,17
08F7  90           SUB B
08F8  38 0B        JR C,0905
08FA  FD BE 22     CP (IY+22)
08FD  DA 35 08     JP C,0835
0900  3C           INC A
0901  47           LD B,A
0902  3E 1F        LD A,1F
0904  91           SUB C
0905  DA AD 0E     JP C,0EAD
0908  C6 02        ADD A,02
090A  4F           LD C,A
090B  FD CB 01 4E  BIT 1,(IY+01)
090F  28 07        JR Z,0918
0911  3E 5D        LD A,5D
0913  91           SUB C
0914  32 38 40     LD (4038),A
0917  C9           RET
0918  ED 43 39 40  LD (4039),BC
091C  2A 10 40     LD HL,(4010)
091F  51           LD D,C
0920  3E 22        LD A,22
0922  91           SUB C
0923  4F           LD C,A
0924  3E 76        LD A,76
0926  04           INC B
0927  2B           DEC HL
0928  BE           CP (HL)
0929  20 FC        JR NZ,0927
092B  10 FA        DJNZ 0927
092D  23           INC HL
092E  ED B1        CPIR
0930  2B           DEC HL
0931  22 0E 40     LD (400E),HL
0934  37           SCF
0935  E0           RET PO
0936  15           DEC D
0937  C8           RET Z
0938  C5           PUSH BC
0939  CD 9E 09     CALL 099E
093C  C1           POP BC
093D  41           LD B,C
093E  62           LD H,D
093F  6B           LD L,E
0940  36 00        LD (HL),00
0942  2B           DEC HL
```

```
0943  10 FB        DJNZ 0940
0945  EB           EX DE,HL
0946  23           INC HL
0947  22 0E 40     LD (400E),HL
094A  C9           RET
094B  F5           PUSH AF
094C  CD 75 09     CALL 0975
094F  30 08        JR NC,0959
0951  FD CB 01 46  BIT 0,(IY+01)
0955  20 02        JR NZ,0959
0957  AF           XOR A
0958  D7           RST 10
0959  0A           LD A,(BC)
095A  E6 3F        AND 3F
095C  D7           RST 10
095D  0A           LD A,(BC)
095E  03           INC BC
095F  87           ADD A,A
0960  30 F7        JR NC,0959
0962  C1           POP BC
0963  CB 78        BIT 7,B
0965  C8           RET Z
0966  FE 1A        CP 1A
0968  28 03        JR Z,096D
096A  FE 38        CP 38
096C  D8           RET C
096D  AF           XOR A
096E  FD CB 01 C6  SET 0,(IY+01)
0972  C3 F5 07     JP 07F5
0975  E5           PUSH HL
0976  21 11 01     LD HL,0111
0979  CB 7F        BIT 7,A
097B  28 02        JR Z,097F
097D  E6 3F        AND 3F
097F  FE 43        CP 43
0981  30 10        JR NC,0993
0983  47           LD B,A
0984  04           INC B
0985  CB 7E        BIT 7,(HL)
0987  23           INC HL
0988  28 FB        JR Z,0985
098A  10 F9        DJNZ 0985
098C  CB 77        BIT 6,A
098E  20 02        JR NZ,0992
0990  FE 18        CP 18
0992  3F           CCF
0993  44           LD B,H
0994  4D           LD C,L
0995  E1           POP HL
0996  D0           RET NC
0997  0A           LD A,(BC)
0998  C6 E4        ADD A,E4
099A  C9           RET
099B  01 01 00     LD BC,0001
099E  E5           PUSH HL
099F  CD C5 0E     CALL 0EC5
09A2  E1           POP HL
09A3  CD AD 09     CALL 09AD
09A6  2A 1C 40     LD HL,(401C)
09A9  EB           EX DE,HL
09AA  ED B8        LDDR
09AC  C9           RET
09AD  F5           PUSH AF
09AE  E5           PUSH HL
09AF  21 0C 40     LD HL,400C
09B2  3E 09        LD A,09
09B4  5E           LD E,(HL)
09B5  23           INC HL
09B6  56           LD D,(HL)
09B7  E3           EX (SP),HL
09B8  A7           AND A
09B9  ED 52        SBC HL,DE
09BB  19           ADD HL,DE
09BC  E3           EX (SP),HL
09BD  30 09        JR NC,09C8
09BF  D5           PUSH DE
09C0  EB           EX DE,HL
09C1  09           ADD HL,BC
09C2  EB           EX DE,HL
09C3  72           LD (HL),D
09C4  2B           DEC HL
09C5  73           LD (HL),E
09C6  23           INC HL
09C7  D1           POP DE
09C8  23           INC HL
09C9  3D           DEC A
09CA  20 E8        JR NZ,09B4
09CC  EB           EX DE,HL
09CD  D1           POP DE
09CE  F1           POP AF
09CF  A7           AND A
09D0  ED 52        SBC HL,DE
09D2  44           LD B,H
09D3  4D           LD C,L
09D4  03           INC BC
09D5  19           ADD HL,DE
09D6  EB           EX DE,HL
09D7  C9           RET
09D8  E5           PUSH HL
09D9  21 7D 40     LD HL,407D
09DC  54           LD D,H
09DD  5D           LD E,L
09DE  C1           POP BC
09DF  CD EA 09     CALL 09EA
09E2  D0           RET NC
09E3  C5           PUSH BC
09E4  CD F2 09     CALL 09F2
09E7  EB           EX DE,HL
09E8  18 F4        JR 09DE
09EA  7E           LD A,(HL)
09EB  B8           CP B
09EC  C0           RET NZ
09ED  23           INC HL
09EE  7E           LD A,(HL)
09EF  2B           DEC HL
09F0  B9           CP C
09F1  C9           RET
09F2  E5           PUSH HL
09F3  7E           LD A,(HL)
09F4  FE 40        CP 40
09F6  38 17        JR C,0A0F
09F8  CB 6F        BIT 5,A
09FA  28 14        JR Z,0A10
09FC  87           ADD A,A
09FD  FA 01 0A     JP M,0A01
0A00  3F           CCF
0A01  01 05 00     LD BC,0005
0A04  30 02        JR NC,0A08
0A06  0E 11        LD C,11
0A08  17           RLA
0A09  23           INC HL
0A0A  7E           LD A,(HL)
0A0B  30 FB        JR NC,0A08
0A0D  18 06        JR 0A15
0A0F  23           INC HL
0A10  23           INC HL
0A11  4E           LD C,(HL)
0A12  23           INC HL
0A13  46           LD B,(HL)
0A14  23           INC HL
0A15  09           ADD HL,BC
0A16  D1           POP DE
0A17  A7           AND A
0A18  ED 52        SBC HL,DE
```

```
0A1A  44           LD B,H
0A1B  4D           LD C,L
0A1C  19           ADD HL,DE
0A1D  EB           EX DE,HL
0A1E  C9           RET
0A1F  FD 46 22     LD B,(IY+22)
0A22  C5           PUSH BC
0A23  CD 2C 0A     CALL 0A2C
0A26  C1           POP BC
0A27  05           DEC B
0A28  18 02        JR 0A2C
0A2A  06 18        LD B,18
0A2C  FD CB 01 8E  RES 1,(IY+01)
0A30  0E 21        LD C,21
0A32  C5           PUSH BC
0A33  CD 18 09     CALL 0918
0A36  C1           POP BC
0A37  3A 05 40     LD A,(4005)
0A3A  FE 4D        CP 4D
0A3C  38 14        JR C,0A52
0A3E  FD CB 3A FE  SET 7,(IY+3A)
0A42  AF           XOR A
0A43  CD F5 07     CALL 07F5
0A46  2A 39 40     LD HL,(4039)
0A49  7D           LD A,L
0A4A  B4           OR H
0A4B  E6 7E        AND 7E
0A4D  20 F3        JR NZ,0A42
0A4F  C3 18 09     JP 0918
0A52  54           LD D,H
0A53  5D           LD E,L
0A54  2B           DEC HL
0A55  48           LD C,B
0A56  06 00        LD B,00
0A58  ED B0        LDIR
0A5A  2A 10 40     LD HL,(4010)
0A5D  CD 17 0A     CALL 0A17
0A60  C5           PUSH BC
0A61  78           LD A,B
0A62  2F           CPL
0A63  47           LD B,A
0A64  79           LD A,C
0A65  2F           CPL
0A66  4F           LD C,A
0A67  03           INC BC
0A68  CD AD 09     CALL 09AD
0A6B  EB           EX DE,HL
0A6C  E1           POP HL
0A6D  19           ADD HL,DE
0A6E  D5           PUSH DE
0A6F  ED B0        LDIR
0A71  E1           POP HL
0A72  C9           RET
0A73  2A 14 40     LD HL,(4014)
0A76  CD 4D 00     CALL 004D
0A79  DF           RST 18
0A7A  FD CB 2D 6E  BIT 5,(IY+2D)
0A7E  C0           RET NZ
0A7F  21 5D 40     LD HL,405D
0A82  22 1C 40     LD (401C),HL
0A85  CD 44 15     CALL 1544
0A88  CD 86 15     CALL 1586
0A8B  38 04        JR C,0A91
0A8D  21 F0 D8     LD HL,D8F0
0A90  09           ADD HL,BC
0A91  DA 9A 0D     JP C,0D9A
0A94  BF           CP A
0A95  C3 B8 14     JP 14B8
0A98  D5           PUSH DE
0A99  E5           PUSH HL
0A9A  AF           XOR A
0A9D  20 20        JR NZ,0ABF
0A9F  60           LD H,B
0AA0  69           LD L,C
0AA1  1E FF        LD E,FF
0AA3  18 08        JR 0AAD
0AA5  D5           PUSH DE
0AA6  56           LD D,(HL)
0AA7  23           INC HL
0AA8  5E           LD E,(HL)
0AA9  E5           PUSH HL
0AAA  EB           EX DE,HL
0AAB  1E 00        LD E,00
0AAD  01 18 FC     LD BC,FC18
0AB0  CD E1 07     CALL 07E1
0AB3  01 9C FF     LD BC,FF9C
0AB6  CD E1 07     CALL 07E1
0AB9  0E F6        LD C,F6
0ABB  CD E1 07     CALL 07E1
0ABE  7D           LD A,L
0ABF  CD EB 07     CALL 07EB
0AC2  E1           POP HL
0AC3  D1           POP DE
0AC4  C9           RET
0AC5  CD A6 0D     CALL 0DA6
0AC8  E1           POP HL
0AC9  C8           RET Z
0ACA  E9           JP (HL)
0ACB  FD CB 01 CE  SET 1,(IY+01)
0ACF  7E           LD A,(HL)
0AD0  FE 76        CP 76
0AD2  CA 84 0B     JP Z,0B84
0AD5  D6 1A        SUB 1A
0AD7  CE 00        ADC A,00
0AD9  28 69        JR Z,0B44
0ADB  FE A7        CP A7
0ADD  20 1B        JR NZ,0AFA
0ADF  E7           RST 20
0AE0  CD 92 0D     CALL 0D92
0AE3  FE 1A        CP 1A
0AE5  C2 9A 0D     JP NZ,0D9A
0AE8  E7           RST 20
0AE9  CD 92 0D     CALL 0D92
0AEC  CD 4E 0B     CALL 0B4E
0AEF  EF           RST 28
0AF0  01 34 CD     LD BC,CD34
0AF3  F5           PUSH AF
0AF4  0B           DEC BC
0AF5  CD F5 08     CALL 08F5
0AF8  18 3D        JR 0B37
0AFA  FE A8        CP A8
0AFC  20 33        JR NZ,0B31
0AFE  E7           RST 20
0AFF  CD 92 0D     CALL 0D92
0B02  CD 4E 0B     CALL 0B4E
0B05  CD 02 0C     CALL 0C02
0B08  C2 AD 0E     JP NZ,0EAD
0B0B  E6 1F        AND 1F
0B0D  4F           LD C,A
0B0E  FD CB 01 4E  BIT 1,(IY+01)
0B12  28 0A        JR Z,0B1E
0B14  FD 96 38     SUB (IY+38)
0B17  CB FF        SET 7,A
0B19  C6 3C        ADD A,3C
0B1B  D4 71 08     CALL NC,0871
0B1E  FD 86 39     ADD A,(IY+39)
0B21  FE 21        CP 21
0B23  3A 3A 40     LD A,(403A)
0B26  DE 01        SBC A,01
0B28  CD FA 08     CALL 08FA
0B2B  FD CB 01 C6  SET 0,(IY+01)
0B2F  18 06        JR 0B37
```

```
0B34  CD 55 0B           CALL 0B55
0B37  DF                 RST 18
0B38  D6 1A              SUB 1A
0B3A  CE 00              ADC A,00
0B3C  28 06              JR Z,0B44
0B3E  CD 1D 0D           CALL 0D1D
0B41  C3 84 0B           JP 0B84
0B44  D4 BB 0B           CALL NC,0BBB
0B47  E7                 RST 20
0B48  FE 76              CP 76
0B4A  C8                 RET Z
0B4B  C3 D5 0A           JP 0AD5
0B4E  CD A6 0D           CALL 0DA6
0B51  C0                 RET NZ
0B52  E1                 POP HL
0B53  18 E2              JR 0B37
0B55  CD C5 0A           CALL 0AC5
0B58  FD CB 01 76        BIT 6,(IY+01)
0B5C  CC F4 13           CALL Z,13F4
0B5F  28 0A              JR Z,0B6B
0B61  C3 D7 15           JP 15D7
0B64  3E 0B              LD A,0B
0B66  D7                 RST 10
0B67  ED 5B 18 40        LD DE,(4018)
0B6B  78                 LD A,B
0B6C  B1                 OR C
0B6D  0B                 DEC BC
0B6E  C8                 RET Z
0B6F  1A                 LD A,(DE)
0B70  13                 INC DE
0B71  ED 53 18 40        LD (4018),DE
0B75  CB 77              BIT 6,A
0B77  28 ED              JR Z,0B66
0B79  FE C0              CP C0
0B7B  28 E7              JR Z,0B64
0B7D  C5                 PUSH BC
0B7E  CD 4B 09           CALL 094B
0B81  C1                 POP BC
0B82  18 E3              JR 0B67
0B84  CD C5 0A           CALL 0AC5
0B87  3E 76              LD A,76
0B89  D7                 RST 10
0B8A  C9                 RET
0B8B  CD C5 0A           CALL 0AC5
0B8E  FD CB 01 C6        SET 0,(IY+01)
0B92  AF                 XOR A
0B93  D7                 RST 10
0B94  ED 4B 39 40        LD BC,(4039)
0B98  79                 LD A,C
0B99  FD CB 01 4E        BIT 1,(IY+01)
0B9D  28 05              JR Z,0BA4
0B9F  3E 5D              LD A,5D
0BA1  FD 96 38           SUB (IY+38)
0BA4  0E 11              LD C,11
0BA6  B9                 CP C
0BA7  30 02              JR NC,0BAB
0BA9  0E 01              LD C,01
0BAB  CD 0B 09           CALL 090B
0BAE  C9                 RET
0BAF  CD F5 0B           CALL 0BF5
0BB2  ED 43 36 40        LD (4036),BC
0BB6  3E 2B              LD A,2B
0BB8  90                 SUB B
0BB9  DA AD 0E           JP C,0EAD
0BBC  47                 LD B,A
0BBD  3E 01              LD A,01
0BBF  CB 28              SRA B
0BC1  30 02              JR NC,0BC5
0BC3  3E 04              LD A,04
0BC5  CB 29              SRA C
0BC7  30 01              JR NC,0BCA
0BC9  07                 RLCA
```

```
0BCA  F5                 PUSH AF
0BCB  CD F5 0B           CALL 0BF5
0BCE  7E                 LD A,(HL)
0BCF  07                 RLCA
0BD0  FE 10              CP 10
0BD2  30 06              JR NC,0BDA
0BD4  0F                 RRCA
0BD5  30 02              JR NC,0BD9
0BD7  EE 8F              XOR 8F
0BD9  47                 LD B,A
0BDA  11 9E 0C           LD DE,0C9E
0BDD  3A 30 40           LD A,(4030)
0BE0  93                 SUB E
0BE1  FA E9 0B           JP M,0BE9
0BE4  F1                 POP AF
0BE5  2F                 CPL
0BE6  A0                 AND B
0BE7  18 02              JR 0BEB
0BE9  F1                 POP AF
0BEA  B0                 OR B
0BEB  FE 08              CP 08
0BED  38 02              JR C,0BF1
0BEF  EE 8F              XOR 8F
0BF1  D9                 EXX
0BF2  D7                 RST 10
0BF3  D9                 EXX
0BF4  C9                 RET
0BF5  CD 02 0C           CALL 0C02
0BF8  47                 LD B,A
0BF9  C5                 PUSH BC
0BFA  CD 02 0C           CALL 0C02
0BFD  59                 LD E,C
0BFE  C1                 POP BC
0BFF  51                 LD D,C
0C00  4F                 LD C,A
0C01  C9                 RET
0C02  CD C9 15           CALL 15C9
0C05  DA AD 0E           JP C,0EAD
0C08  0E 01              LD C,01
0C0A  C8                 RET Z
0C0B  0E FF              LD C,FF
0C0D  C9                 RET
0C0E  FD 46 22           LD B,(IY+22)
0C11  0E 21              LD C,21
0C13  CD 18 09           CALL 0918
0C16  CD 9B 09           CALL 099B
0C19  7E                 LD A,(HL)
0C1A  12                 LD (DE),A
0C1B  FD 34 3A           INC (IY+3A)
0C1E  2A 0C 40           LD HL,(400C)
0C21  23                 INC HL
0C22  54                 LD D,H
0C23  5D                 LD E,L
0C24  ED B1              CPIR
0C26  C3 5D 0A           JP 0A5D
0C29  8B                                   8D 2D   7F 81 49 75
      5F 40 42 2B   17 1F 37 52   45 0F 6D 2B   44 2D 5A 3B
      4C 45 0D 52   5A 4D 15 6A   01 14 02 06   00 81 0E 06
      DE 05 AB 0D   06 00 B5 0E   00 DC 0C 00   D8 0E 04 14
      06 DF 06 05   B9 0D 04 00   2E 0E 05 CF   0A 01 00 E9
      0F 05 05 14   05 6A 0D 00   C3 03 03 AF   0E 03 30 07
      06 1A 06 00   92 0E 03 6C   0E 05 40 03   05 F6 02 00
      7C 0E 00 96   14 00 2A 0A   06 1A 06 00   AF 0B 06 1A
      06 00 AF 0B   00 0E 0C 06   00 2F 0F 00   28 0F 00 20
      0F 00 69 0B   05 CB 0A 03   2C 07
```

SOLUTIONS TO EXERCISES

Chapter 1

1(a)

```
10 FOR X = 63 TO 0 STEP -1  ... use 59 for 1K ZX81's
20 FOR Y = 43 TO 0 STEP -1
30 PLOT X,Y
40 NEXT Y
50 NEXT X
```

1(b)

```
10 FOR X = 10 TO 30
20 FOR Y = 5 TO 25
30 PLOT X,Y
40 NEXT Y
50 NEXT X
```

1(c) For 16K Machines:–

```
 10 FOR X = 0 TO 63
 20 PLOT X,0
 30 GO SUB 500
 40 UNPLOT X,0
 50 NEXT X
 60 FOR Y = 0 TO 43
 70 PLOT 63,Y
 80 GO SUB 500
 90 UNPLOT 63,Y
100 NEXT Y
110 FOR X=63 TO 0 STEP -1
120 PLOT X,43
130 GO SUB 500
140 UNPLOT X,43
150 NEXT X
160 FOR Y=43 TO 0 STEP -1
170 PLOT 0,Y
180 GO SUB 500
190 UNPLOT 0, Y
200 NEXT Y
210 STOP
500 PAUSE 10
510 POKE 16437,255
520 RETURN
```

For 1K ZX81's the boundaries of the display will need to be reduced.

Alternatively try this. (Substitute Y=15 for Y=Ø for 1K):

```
10 LET X=0
20 LET Y=0
30 PLOT X,Y
40 PAUSE 10          ) or  40 FOR A=1 TO 20
50 POKE 16437,255)         50 NEXT A
60 UNPLOT X,Y
70 LET X=X+(Y=0)-(Y=43)-(X=63 AND Y=0) + (X=0 AND
   Y=43)
80 LET Y=Y-(X=0)+(X=63)-(Y=43 AND X=63) + (Y=0 AND
   X=0)
90 GO TO 30
```

1(d)

```
10 FOR A=0 TO 90 STEP 5
20 LET M=TAN(A*2*PI/360)
30 FOR X=0 TO 63 (use 40 for 1K)
40 LET Y=INT (M*X)
50 IF Y>43 THEN GO TO 80
60 PLOT X,Y
70 NEXT X
80 NEXT A
```

1(e) A line through the point (32,22) with gradient m can be calculated since the slope $\frac{y-22}{x-32} = m$

so $y = m(x-32)+22$

Use values of m from −5 to 5 in steps of one half

```
10 FOR M=-5 TO 5 STEP 0.5
20 FOR X=0 TO 63
30 LET Y=INT(M*(X-32)+22)
40 IF Y>43 OR Y<0 THEN GO TO 60
50 PLOT X,Y
60 NEXT X
70 NEXT M
```

1(f) Equation is $x^2 + y^2 = 40^2$

$$\text{so } x = 40\cos\theta$$
$$\text{and } y = 40\sin\theta$$

and we PLOT it for the angle θ between 0° and 90°

```
10 FOR Q = ØTO 9Ø
2Ø LET P=Q*PI/18Ø
3Ø PLOT 4Ø*COS P,4Ø*SIN P
4Ø NEXT Q
```

Chapter 2

2(a) Add the following instruction

```
55 IF A$(I)="–" OR A$(I)="b" THEN GO TO 7Ø
```

2(b)

```
 1Ø DIM S$(4Ø)
 2Ø PRINT "ENTERbSENTENCE:"
 3Ø INPUT S$
 4Ø PRINT S$
 5Ø FOR I=1 TO 37
 6Ø IF S$(I TO I+2)="THE" THEN GO TO 1ØØ
 7Ø NEXT I
 8Ø PRINT "DOESbNOTbCONTAINbTHE."
 9Ø STOP
1ØØ PRINT "DOESbCONTAINbTHE"
```

Notice this accepts any word containing, T,H,E, e.g. PATHETIC

Chapter 3

3(a) No solution specified: the subject area is open to the reader's choice.

3(b)

```
71 LET V=Ø
72 FOR J=1 TO 8
73 IF CODE (W$(I,J))>128 THEN LET V=V+1
74 NEXT J
75 IF V<>1 THEN GO TO 7Ø
```

V counts the number of inverse characters in the word.

3(c) ABS is included so that the **relative** values of N and R are compared effectively, i.e. it is unimportant which is the larger. Without ABS we have for example:

```
190 IF N-R>5 THEN GO TO 220
```

If R was 50 and N was 44 the test would not be satisfied since −6 is not greater than 5.

3(d) Try it and see!

3(e)

```
30 LET A$(I)=CHR$(INT(RND*11)+28)
```

3(f) There is not a way of crashing the program at the input stage that we have found!

3(g) Add the following line

```
105 IF N=M AND D>SQR M THEN GO TO 120
```

3(h) Line 260 terminates with an error if DF is zero. Add the following:

```
253 IF DF<>0 THEN GO TO 260
255 PRINT "NObSOLUTIONbPOSSIBLE"
257 STOP
```

3(i) Line 360 detects a response which is shorter than the answer keyword, and which is therefore obviously wrong.

APPENDIX
PROGRAM DESIGN AND DEVELOPMENT

Introduction

This section has been written to show the reader how a games program has been built up from first ideas into a fully working and documented program.

The writing of programs can generally be split into a number of steps:

1. Defining the problem
2. Outlining the solution
3. Selecting and representing algorithms
4. Coding (or writing the program)
5. Debugging
6. Testing and validating
7. Documentation
8. Maintaining the program

It is vital that a considerable amount of planning and design for a program takes place before the user touches the keyboard. In particular the aim and format of the program must be clearly specified, since ambiguities at this stage will cause problems later. When defining the method of solution (or **algorithm**) is it helpful to write this down as a series of separate steps in a block-structured form, as shown in earlier chapters. Some programmers like to use **flowcharts** (example later) but the author thinks these are not vital if the algorithm is written down in a structured way. All later stages of the implementation of the program are based upon this stage and it can be helpful to specify the names and meanings of variables here. Certainly a list of variables must be kept as the program is written to avoid confusion or duplication of names. Even when tracking down errors or program **bugs**, the structured method can be traced through to show any logic errors. Although it can be frustrating for the user with a good idea for a program to wait for an hour or so following this method before getting on the ZX81, he will find it saves a great deal of time later: there will be much fewer errors and those present will be easier to find.

This approach, while certainly the best, does present problems for those beginning to program, since such users will not have all their ideas at the beginning of the design process. The remainder of this section represents an alternative of gradually improving upon an initial simple program.

Sample Specification

Step One, defining the problem means "What is the Program to do" or "What is its specification". Here is the specification of the program which will be covered in this section:

"The program is to draw a large block on the screen which represents a thick dungeon wall. A prisoner under the user's control has to dig himself out from one end to the other. However parts of the wall are of made of hard rock, which he must dig round. If he takes too long a Warder will come looking for him and if he is found he will be taken back and the tunnels he has dug filled in. The object is for the prisoner to escape."

Those people without 16K expansions will realise that (without using machine code) it will not be possible for their ZX81's to handle such a program. Even so the first part of this section is just as applicable to 1K as to 16K, so continue reading.

The specification has given us three main problems:

1. We have to be able to move a character representing the prisoner round the screen.
2. Parts of the screen have to be designated "No-Go" areas which the prisoner must go round, and
3. We have to make a second character follow the paths made by the "Prisoner" while looking for him.

PROBLEM ONE – MOVING A CHARACTER ROUND THE SCREEN

As described earlier in the book there are two ways of printing a character on the screen, "PRINT AT" and "PLOT". We will use "PRINT AT" because any character can be displayed by this statement. Since it is not necessary to input the co-ordinates for every move and since we do not want the program to stop while waiting for an input we will use the 'INKEY$" statement. The following program shows briefly how this can be done.

```
 5  REM PROGRAM 1
10  LET X=5
20  LET Y=1
30  IF INKEY$="8" THEN LET Y=Y+1
40  PRINT AT X,Y;"P"
50  GO TO 30
```

Y is the horizontal position which is incremented everytime the "Right Arrow" or "8" key is pressed thus drawing a line towards the right. Note that the line continues for as long as the "Right Arrow" key is pressed. If we can move in one direction like this then we can move in any direction by incorporating the other arrow keys in the program. One fault with the above program is that once the line reaches a certain length it crashes with an error message B/40 which means that the value of "Y" in line 40 is too large. The computer has tried to draw off the screen,to prevent this happening insert the following line:

```
35  IF Y>30 THEN LET Y=Y-1
```

The line will now no longer be drawn past column 30.

When the above technique is used to draw a line in all directions it becomes a very versatile method of drawing on the screen. The following "Sketcher" program demonstrates this. It makes the ZX81 imitate a childs Etch-A-Sketch machine. Movement and character changing are shown below:

5–8	Left, Down, Up, Right
D	Change character to a dot
B	Change character to a blank
S	Scroll whole picture
A	Change character back to black square
C	Press to insert your own character

```
 5 REM SKETCHER PROGRAM
10 LET X=10
20 LET Y=12
30 LET P$=CHR$(128)
40 LET X=X-(INKEY$="7")+(INKEY$="6")
50 LET Y=Y-(INKEY$="5")+(INKEY$="8")
60 IF INKEY$="C" THEN GO SUB 300
70 IF INKEY$="A" THEN LET P$=CHR$(128)
80 IF INKEY$="D" THEN LET P$="."
```

```
 90 IF INKEY$="B" THEN LET P$="b"
100 IF INKEY$="S" THEN SCROLL
110 IF Y>31 THEN LET X=X+1
120 IF Y>31 THEN LET Y=0
130 IF Y=0 THEN LET Y=Y+1
140 IF X=0 THEN LET X=X+1
150 IF X>20 THEN LET X=0
160 IF X=0 THEN LET Y=Y+1
170 PRINT AT X,Y;P$
180 GO TO 40
300 PRINT AT 1,1;"CHARACTER="
310 INPUT P$
320 PRINT AT 1,1;"bbbbbbbbb"
330 RETURN
```

Lines 10 to 30	Sets up coordinates and character to be printed.
Lines 40 and 50	Change the character co-ordinates, compare with Line 30 in Program 1.
Lines 60 to 90	Allow character to be changed.
Lines 110 to 160	Stop the character from going off the screen.
Lines 300 to 330	Allow the user to change the character to any other character or string.

We have not totally achieved our objective as the Sketcher Program and Program 1 are drawing lines round the screen rather than moving a character. To create the illusion of movement we must erase the character everytime it moves. To do this we must store the old co-ordinates and print a blank on them. Insert the following lines into Program 1:—

```
25 LET S=X
27 LET T=Y
45 PRINT AT S,T;"b"
50 GO TO 25
```

Note how the character flashes — this is because it is constantly rubbing itself out. Try and work out how to stop the flashing (answer at the end of this section) and try to incorporate this technique into the Sketcher Program.

The two programs above both fit into a 1K ZX81 although memory full errors may occur with the Sketcher Program as the screen begins to fill up. For the second problem below (defining parts of the screen as No-Go areas) a 16K expansion must be used.

PROBLEM TWO: DEFINING "NO-GO" AREAS ON SCREEN

By the expression "No-Go Areas" is meant a part of the screen which a program such as Sketcher cannot draw on and must go round to pass. This means that the ZX81 must know what is being displayed on the screen. The easiest way of doing this is to store the screen contents in an array. This is why a 16K expansion is needed. The array will be dimensioned as 21x31 (the number of "PRINT AT" positions) so that every time a character is printed on the screen a corresponding digit should be placed at the correct position in the array. For example if a "3" is printed on the screen at 5,7 then a "3" is stored in the array at 5,7.

```
PRINT AT X,Y;"3"
LET A(X,Y)=3
```

To create No-Go areas on the screen therefore all we have to do is place values in the array. If we try to draw a character in a No-Go screen position then we have to move the character back to its old position.

This BLOCK program gives a demonstration of the No-Go Areas, as well as a moving (self-erasing) character. Move the character by using the "Arrow" keys.

```
  5 REM BLOCK PROGRAM
 10 DIM A(21,31)
 20 FOR S=1 TO 21
 30 FOR T=1 TO 31
 40 LET X=INT(RND*10+1)
 45 IF X>8 THEN PRINT AT S,T;"■"
 50 IF X>8 THEN LET A(S,T)=7
 60 NEXT T
 70 NEXT S
100 LET X=10
110 LET Y=1
120 LET S=X
130 LET T=Y
140 LET X=X-(INKEY$="7")+(INKEY$="6")
150 LET Y=Y-(INKEY$="5")+(INKEY$="8")
```

```
16Ø IF A(X,Y)=7 THEN GO SUB 2ØØ
17Ø PRINT AT X,Y;"O"
18Ø PRINT AT S,T;"b"
19Ø GO TO 12Ø
2ØØ LET X=S
21Ø LET Y=T
22Ø RETURN
```

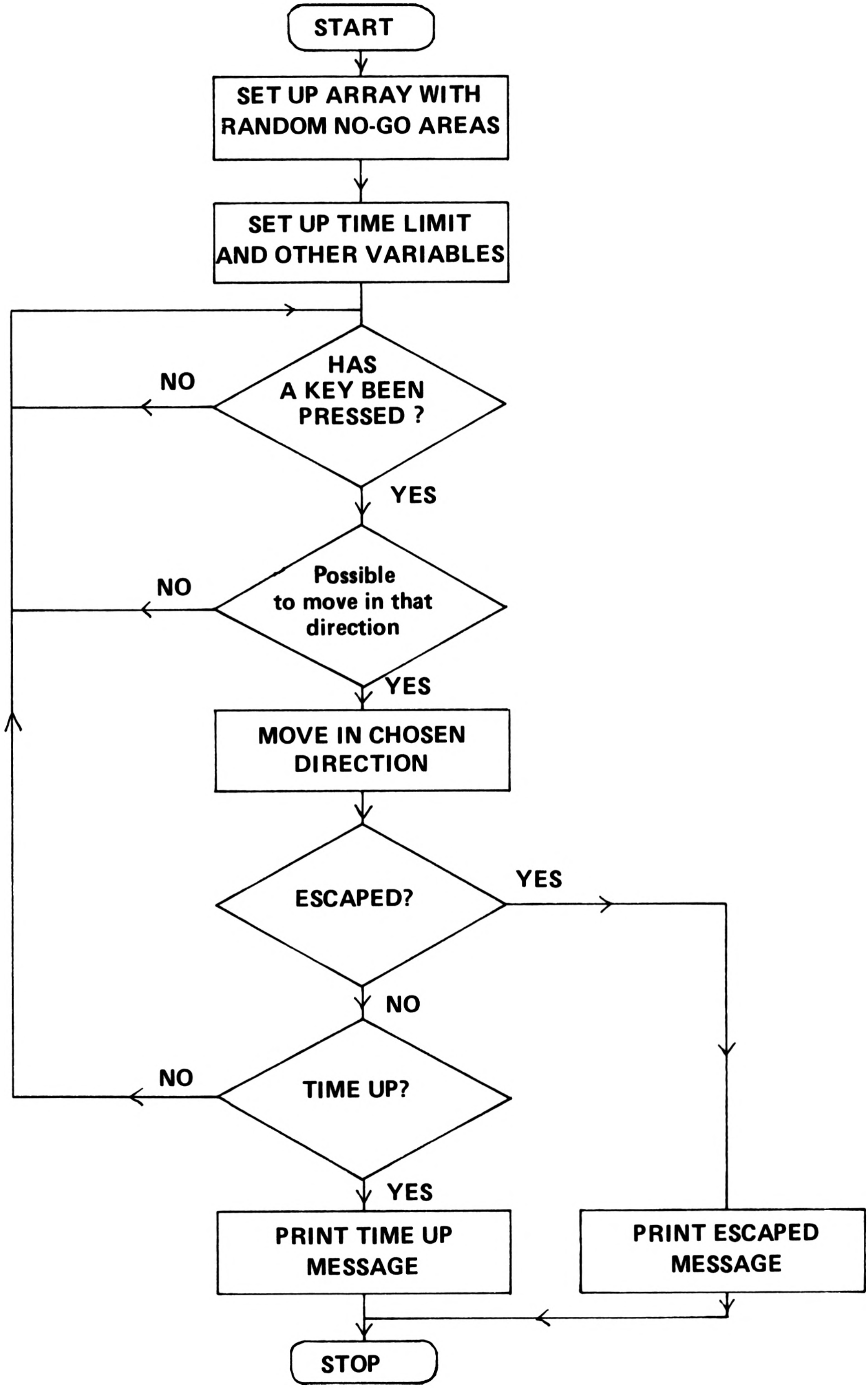
This program produces a screen full of random black squares. Once the flashing "O" has appeared, move about by using the arrow keys. No matter how hard you try you will be unable to move the "O" through a black square. This is how the program works.

Line 1Ø	Sets up the two dimensional array.
Lines 2Ø to 7Ø	Fills randomly chosen parts of the array with 7 and prints out the corresponding position on the screen.
Lines 1ØØ to 11Ø	Set up the starting position of the flashing "O".
Lines 12Ø to 13Ø	Store the previous position of the flashing "O".
Lines 14Ø to 15Ø	Input the new position for the "O" to go to.
Line 16Ø	Finds out whether the new position of the "O" is a No-Go Area by looking at the corresponding position in the array.
Lines 17Ø to 19Ø	Print the "O" and erase the old "O".
Lines 2ØØ to 22Ø	Return the flashing "O" to its old position if it is trying to pass a No-Go Area.

An interesting alteration to liven this program up can be made as follows:—

```
2ØØ LET X=S–2
21Ø LET Y=T–2
```

Now everytime the flashing "O" reaches a No-Go Area it will bounce away from it. As we now have the two main techniques which form the basis of the game program we can now begin work on it. The diagram on the next page shows a general algorithm for the game in the form of a flowchart. It is around this that we shall write the Game Program.


START
SET UP ARRAY WITH RANDOM NO-GO AREAS
SET UP TIME LIMIT AND OTHER VARIABLES
HAS A KEY BEEN PRESSED ?
NO
YES
Possible to move in that direction
NO
YES
MOVE IN CHOSEN DIRECTION
ESCAPED?
YES
NO
TIME UP?
NO
YES
PRINT TIME UP MESSAGE
PRINT ESCAPED MESSAGE
STOP

This flowchart is not an algorithm for the complete game because we want a "Warder" to be able to chase the "Prisoner" through the "tunnels" and capture him. However this part of the program can be added later at the "Print Time Up" box. Box 1 carries out the same functions as lines 1Ø to 1ØØ in the block program and can be broken down into the following sections.

a) Set up array
b) Set up initial variables (co-ordinates etc.)
c) Fill array with random flags (No-go areas)
d) Print out picture
e) Set up time limit

Up to this point we have reached Step 3, in the program — writing procedure, that is we have:—

DEFINED THE PROBLEM, this was our specification.

OUTLINED THE SOLUTION, the techniques needed to write the program, i.e. character movement and No-go Areas.

SELECTED AND REPRESENTED ALGORITHMS, such as the flow-chart and the sketch and block programs.

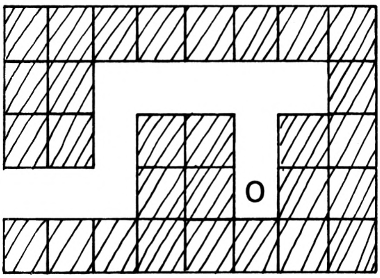
We have also carried out some coding, debugging and documentation in the process of outlining the solution. Coding of the actual game program can now be done as all of the boxes in the flowchart have been covered already.

```
 1Ø REM ESCAPE GAME
 2Ø REM INITIALISE
 3Ø FAST
 4Ø DIM A(2Ø,22)
 5Ø LET X=1Ø
 6Ø LET Y=2
 7Ø LET C=Ø
 8Ø LET A$="O"
 9Ø REM FILL ARRAY
1ØØ FOR I=1 TO 1ØØ
11Ø LET V=INT(RND*2Ø+1)
12Ø LET D=INT(RND*2Ø+1)
13Ø LET A(V,D)=7
14Ø NEXT I
15Ø REM PRINT PICTURE
```

```
16Ø SLOW
17Ø CLS
18Ø FOR I=1 TO 2Ø
19Ø PRINT "bb▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒
    ▒"
2ØØ NEXT I
21Ø REM SETUP TIME LIMIT
22Ø LET TIME = INT(RND*2ØØ+1)
23Ø REM MOVING PRISONER
24Ø LET C=C+1
25Ø IF C>TIME THEN GO TO 5ØØ
26Ø LET K=X
27Ø LET P=Y
28Ø LET X=X-(INKEY$="7")+(INKEY$="6")
29Ø LET Y=Y-(INKEY$="5")+(INKEY$="8")
295 IF Y>2Ø THEN GO TO 4ØØ
3ØØ IF A(X,Y)=7 THEN LET X=K
31Ø IF A(X,Y)=7 THEN LET Y=P
32Ø PRINT AT K,P;"b"
33Ø LET A(X,Y)=8
34Ø PRINT AT X,Y;A$
35Ø GO TO 23Ø
4ØØ PRINT AT 1,2Ø;"YOUbHAVE"
41Ø PRINT AT 3,2Ø;"ESCAPED"
42Ø GO TO 7ØØ
5ØØ PRINT AT 1,2Ø;"YOUbARE"
51Ø PRINT AT 3,2Ø ;"CAUGHT"
7ØØ STOP
```

When this program is run the screen will clear for about five seconds while the initialisation and array-filling take place (lines 1Ø to 14Ø). A large black square is then drawn on the left of the T.V. screen with a flashing "O" at the left edge. The flashing "O" represents the prisoner and the black square shows the ground through which he must dig to make his escape. Escape is achieved when he reaches the right hand side. You may have noticed that this program uses a slightly different method of filling the array than does the block program. In this program the density of No-Go areas can be chosen (here 1ØØ) by the size of the FOR NEXT loop in line 1ØØ. The "Tunnel" effect is created by the printing of blanks in line 32Ø which also causes the "O" to flash. Try making changes to the program, for example to the density of No-Go areas, the time limit, or adding boundaries to the top, bottom and left of the screen so that the program does not crash with a 3/300 error if you go off the edge.

This game provides a good springboard for further expansions, however we said that it was our aim to incorporate a "Warder" to capture the prisoner. The warder has to either follow the exact course taken by the prisoner or work his way through the tunnels by following the array. Let us look at the second method first. The array used by this program so far contains 3 numbers; Ø, 7, 8, which correspond to the screen display as shown below.



7	7	7	0	0	7	0	0
0	0	8	8	8	8	8	7
0	7	8	7	7	8	7	0
8	8	8	0	0	8	0	0
0	0	0	0	0	0	0	7

SCREEN DISPLAY

8 = Passage taken by "Prisoner"
7 = No go areas
0 = Areas which can be passed through

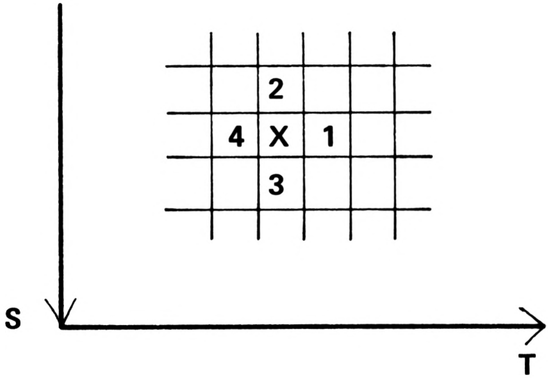
As the ZX81 will use the array to locate the position of the "Prisoner" the array must show where he is! This can be done by adding the line.

```
245 IF C>TIME THEN LET A(X,Y) = 9
```

Now as soon as the prisoners time is up his position is marked in the array by a "9".

The ZX81 (or warder) can now search from a starting position along the trail of "8"s in the array for the "9" (prisoners position). The computer not only has to search for the "9" but also "8"s so that it can follow a route made by the prisoner. To do this the computer has to search for an "8" or "9" in the squares adjacent to its present position.

Assume the "Warder" is at square X – he will search in the following way:



So to search from	X to 1	T = T + 1
	1 to 2	S = S – 1 & T = T – 1
	2 to 3	S = S + 2
	3 to 4	T = T – 1 & S = S – 1

So that when the ZX81 has found an adjacent "8" it moves onto it and starts the search procedure again. Because this order of searching has been chosen, the "Warder" will tend to move right and up, rather than down and left. The coding to do this will look like this

```
 5ØØ REM CHASING WARDER
 51Ø LET D = 7
 52Ø LET S=1Ø
 53Ø LET T=1
 54Ø LET F=S
 55Ø LET G=T
 56Ø LET T=T+1
 57Ø IF A(S,T)<D THEN GO TO 1ØØØ
 58Ø LET S=S–1
 59Ø LET T=T–1
 6ØØ IF A(S,T)>D THEN GO TO 1ØØØ
 61Ø LET S=S+Z
 62Ø IF A(S,T)>D THEN GO TO 1ØØØ
 63Ø LET S=S–1
 64Ø LET T=T–1
 65Ø IF A(S,T)>D THEN GO TO 1ØØØ
 66Ø GO TO 56Ø
1ØØØ PRINT AT S,T;"X"
```

```
1010 PRINT AT F,G;"b"
1020 IF A(S,T)=9 THEN GO TO 1100
1030 GO TO 540
1100 PRINT AT 1,23;"GOT"
1110 PRINT AT 3,23;"YOU"
```

Because the same order of search is used by the "Warder" (right, up, down, left) it will occasionally become trapped. To prevent this from happening the search order can be randomised, though the "Warder" should be more inclined to move forward than to move backwards.

INDEX

(Program names in Italics)

If you have a Sinclair ZX81 and want to use it to its full potential then, as the experts have all agreed, this is the book for you. It contains detailed guidelines and documented programs in the areas of gaming, information retrieval and education, as well as a unique listing of the 8K ROM for machine code applications.

'Far and away the best . . . once again Linsac has produced the book for the serious end of the market'. – *Your Computer, November 1981.*

'The ZX81 Companion **is a most professional product . . . with many good illustrative programs, tips and warnings'.** – *Education Equipment, October 1981.*

'Bob Maunder's attempt to show meaningful uses of the machine is brilliantly successful . . . thoughtfully written, detailed and illustrated with meaningful programs . . . To conclude – the book is definitely an outstandingly useful second step for the ZX81 user'. – *Educational ZX80/81 Users' Group Newsletter, September 1981.*

Bob Maunder has been involved in the ZX series of microcomputers since he acquired the first ZX80 kit in March 1980, and he is co-author of Linsac's *'The ZX80 Companion'.* He holds a MSc in Computer Science from Birmingham University and is Head of Computing at Hartlepool College, where he pioneered the use of the ZX80 in education.

UK Price £7.95
LINSAC
ISBN 0 907211 01 1

THE ZX81 COMPANION

Bob Maunder

Linsac

AMSTRAD CPC

OCR

MÉMOIRE ÉCRITE
MEMORY ENGRAVED
MEMORIA ESCRITA

ACME https://acpc.me/

[FRA] Ce document a été préservé numériquement à des fins éducatives et d'études, et non commerciales.
[ENG] This document has been digitally preserved for educational and study purposes, not for commercial purposes.
[ESP] Este documento se ha conservado digitalmente con fines educativos y de estudio, no con fines comerciales.